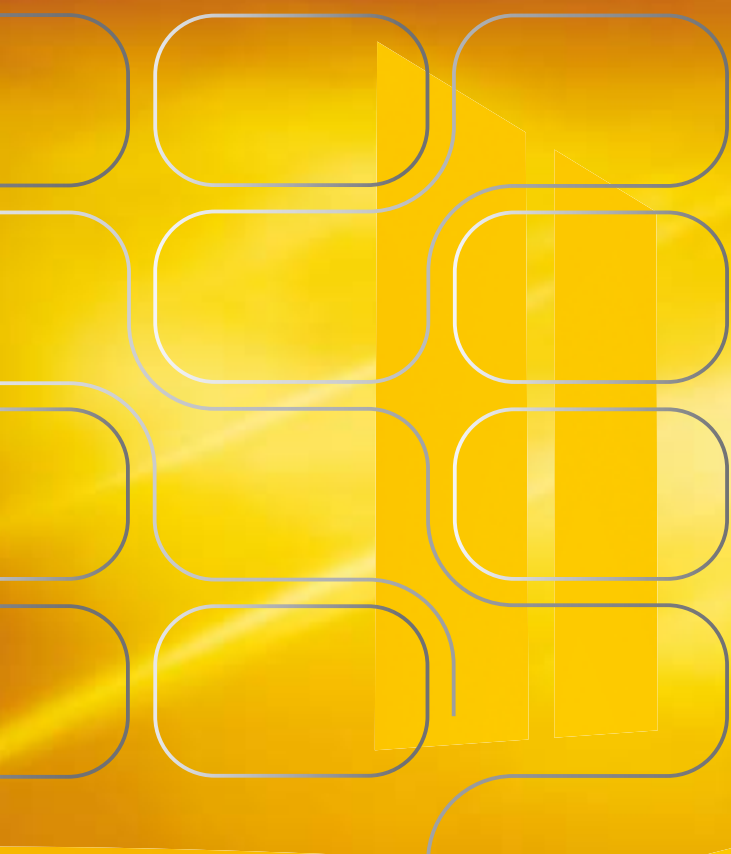


Luboslav Lacko

# Visual Studio Tools for Office



# VISUAL STUDIO TOOLS FOR OFFICE

Luboslav Lacko

*Autor je předním slovenským odborníkem na databázové technologie a programování s více než desetiletou praxí ve vývoji databázových aplikací. Pracuje ve Vojenském technickém ústavu v Liptovském Mikuláši a působí jako školitel a konzultant. Publikuje články především o programování, databázích a data miningu; je autorem několika knih, například Business Intelligence na SQL Serveru 2005, ASP.NET 2.0 hotová Řešení.*

# OBSAH

Kapitola 1: Visual Studio Tools for Office .....	3
Kapitola 2: Základné princípy a architektúra VSTO aplikácie.....	6
Kapitola 3: Vytvorenie VSTO projektov v Exceli .....	9
Kapitola 4 Vytvorenie VSTO projektov vo Worde.....	20
Kapitola 5: Panel ovládacích prvkov.....	33
Kapitola 6: Práca s údajmi v databázach .....	39
Kapitola 7: Práca s objektami a dokumentami.....	52
Kapitola 8: Vytvorenie VSTO projektov v Outlooku.....	60
Kapitola 9: Šírenie VSTO aplikácie .....	73

# KAPITOLA 1: VISUAL STUDIO TOOLS FOR OFFICE

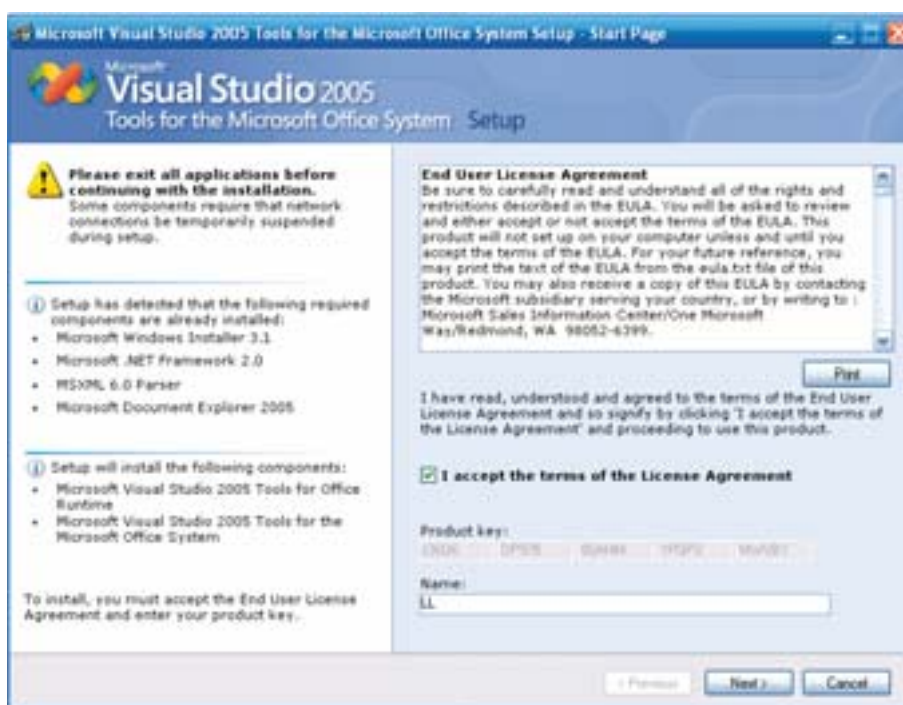
Vývojové prostredie Microsoft Visual Studio 2005 Tools for the Microsoft Office System umožňuje efektívny vývoj aplikácií na báze balíka Office Systems. Ako programovacie jazyky je možné použiť Visual Basic 2005 a Microsoft Visual C#. Vývojové prostredie Visual Studio 2005 Tools Office (VSTO) slúži pre vývoj „dokumentovo – centrických“ aplikácií využívajúcich riadený kód (managed code). Riešenia je možné podľa povahy dokumentov vytvárať pre programy Excel, Word a s prídavným add-ins modulom aj pre Outlook. Princiálne je v rámci VSTO aplikácie možné realizovať prakticky akúkoľvek funkcionality, ktorú dokážeme realizovať vo „Win Forms“ aplikáciách, prípadne ak potrebujeme pracovať s dokumentami balíka Office, je možné klasickú Win Forms aplikáciu pomerne jednoducho prekonvertovať na VSTO, stačí obľadacie prvky z formulárov umiestniť do modálnych alebo nemodálnych dialógových okien

## Inštalácia

Visual Studio 2005 Tools for Office inštalujeme ako doplnok k Visual Studiu 2005, prípadne ako samostatný produkt, ktorý obsahuje vlastné prostredie Visual Studia a prekladače jazykov Visual Basic 2005 a C# 2.0. V prvej etape inštalácie sa vykoná kontrola, či sú k dispozícii požadované softvérové technologické moduly (Windows Installer 3.0, .NET Framework 2.0, Office...)

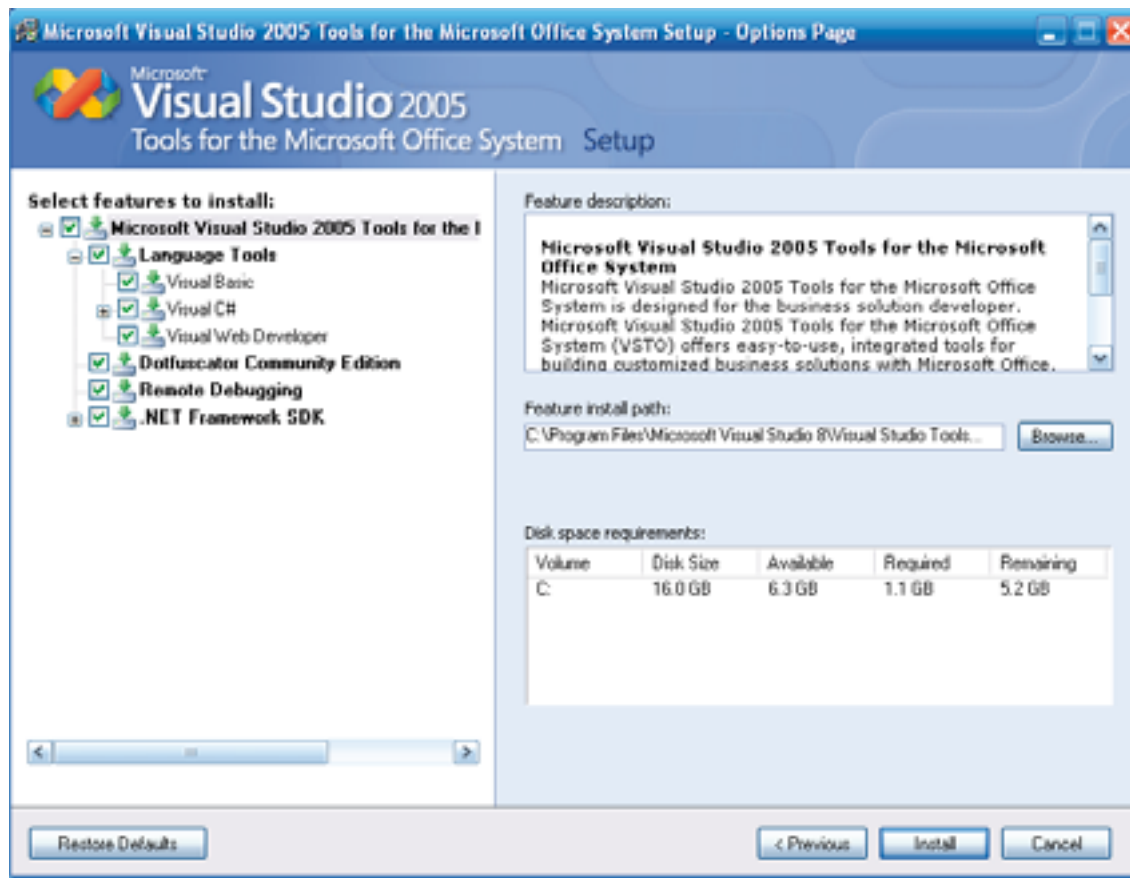


Visual Studio Tools for Office 2005 – úvodný dialóg inštalácie



Zoznam pomocných komponentov, ktoré budú nainštalované

Následne si môžeme vybrať, v akom programovacom jazyku budeme VSTO aplikácie vytvárať. Nakoľko .NET jazyky Visual Basic a Visual C# sú si dosť podobné a v budúcnosti môžeme byť postavení pred úlohu vybudovať VSTO aplikáciu ako následnú verziu už existujúceho kódu, odporúčame nainštalovať podporu všetkých programovacích jazykov



Visual Studio Tools for Office 2005 – voliteľná konfigurácia

Z hľadiska výberu najvhodnejšieho programovacieho jazyka sa pokúsime potvrdiť, alebo vyvrátiť ešte jeden mýtus, podľa ktorého by malo byť výhodnejšie je používať Visual Basic. Je pravda, že niektoré konštrukcie sú v tomto programovacom jazyku jednoduchšie. Vyplýva to z nadväznosti na technológiu VBA (Visual Basic for Application), no nie sú tu pre C# žiadne obmedzenia a teda ani žiadny dôvod pre migráciu z C# na Visual Basic. Pre ukážku sme vybrali fragment kódu pre kontrolu pravopisu. V jazyku Visual Basic vystačíme pre tento účel s pomerne jednoduchým a stručným kódom

```

, Visual Basic
Friend Sub SpellCheckString()
    , Retazec pre chyby.
    Dim str As String = „Chyby budu tu.“
    If ThisApplication.CheckSpelling(str) Then
        MessageBox.Show(String.Format(„Bezchybný retazec „{0}““, str))
    Else
        MessageBox.Show(String.Format(„{0}“ obsahuje chyby“, str))
    End If
End Sub

```

Pre realizáciu tej istej funkcionality potrebujeme oveľa rozsiahlejší kód

```
// C#
public void SpellCheckString()
{
    // Retazec pre chyby.
    string str = „ Chyby budu tu.“;

    Object CustomDictionary = Type.Missing;
    Object IgnoreUppercase = Type.Missing;
    Object MainDictionary = Type.Missing;
    Object CustomDictionary2 = Type.Missing;
    Object CustomDictionary3 = Type.Missing;
    Object CustomDictionary4 = Type.Missing;
    Object CustomDictionary5 = Type.Missing;
    Object CustomDictionary6 = Type.Missing;
    Object CustomDictionary7 = Type.Missing;
    Object CustomDictionary8 = Type.Missing;
    Object CustomDictionary9 = Type.Missing;
    Object CustomDictionary10 = Type.Missing;

    // Odovzdavanie parametrov pre VBA
    if ( ThisApplication.CheckSpelling(str, ref CustomDictionary,
        ref IgnoreUppercase, ref MainDictionary, ref CustomDictionary2,
        ref CustomDictionary3, ref CustomDictionary4,
        ref CustomDictionary5, ref CustomDictionary6,
        ref CustomDictionary7, ref CustomDictionary8,
        ref CustomDictionary9, ref CustomDictionary10) )
    {
        MessageBox.Show(String.Format(„Bezchybný retazec \"{0}\"“, str));
    }
    else
    {
        MessageBox.Show(
            String.Format(„\ \"{0}\" obsahuje chyby“, str));
    }
}
```

Skôr než pristúpime k šíreniu VSTO aplikácií na klientské počítače, musíme si uvedomiť, že tieto aplikácie pre svoj beh vyžaduje inštaláciu podporných knižníc a špeciálne nastavenie prístupových práv. Umiestnenie VSTO aplikácie do klientského prostredia je témou poslednej – desiatej kapitoly.

# KAPITOLA 2: ZÁKLADNÉ PRINCÍPY A ARCHITEKTÚRA

## VSTO APLIKÁCIE

VSTO aplikácie pracujú na princípe „Code Behind“, to znamená okrem dokumentu príslušného programu Office (Word, Excel...) je súčasťou VSTO aplikácie jeden alebo viaceré súbory obsahujúce výkonný kód. No nielen to. Použiť môžeme aj širokú paletu WinForms ovládacích prvkov, ktoré môžeme rozmiestniť nielen na ploche ovládacích panelov ale aj priamo na ploche dokumentov. Vizualne aj nevizualne ovládacie prvky podobne ako u klasických aplikácií rozmiestňujeme buď v etape návrhu a vývoja pomocou vizuálnych nástrojov z toolboxu, alebo ich môžeme, podobne ako vo „Win Forms“ aplikáciách vytvárať dynamicky počas behu aplikácie. K dispozícii máme aj nové ovládacie prvky „host controls“, ktoré obohacujú klasické Wordovské, alebo Excelovské objekty o .NET funkcionality. Ovládacie prvky môžeme vytvárať buď v etape návrhu, alebo dynamicky počas behu aplikácie.

### Objektový model wordovej aplikácie

Podobne ako všetky moderné technologické platformy pre podporu vývoja aplikácií aj VSTO aplikácie sú vybudované na hierarchickom objektovom modeli. Pre efektívny vývoj s využitím dostupných prvkov je potrebné poznať túto objektovú štruktúru aspoň v hrubých rysoch. Podobne ako u klasických objektovo orientovaných aplikáciách sa snažíme pomocou objektov vyjadriť vzťahy, zákonitosti a hierarchiu vonkajšieho sveta. Základné objekty pre budovanie VSTO aplikácie sa snažia vyjadrovať základnú funkcionality príslušných Office aplikácií (Word, Excel, Outlook), aby ju bolo možné efektívne využívať v aplikáciách.

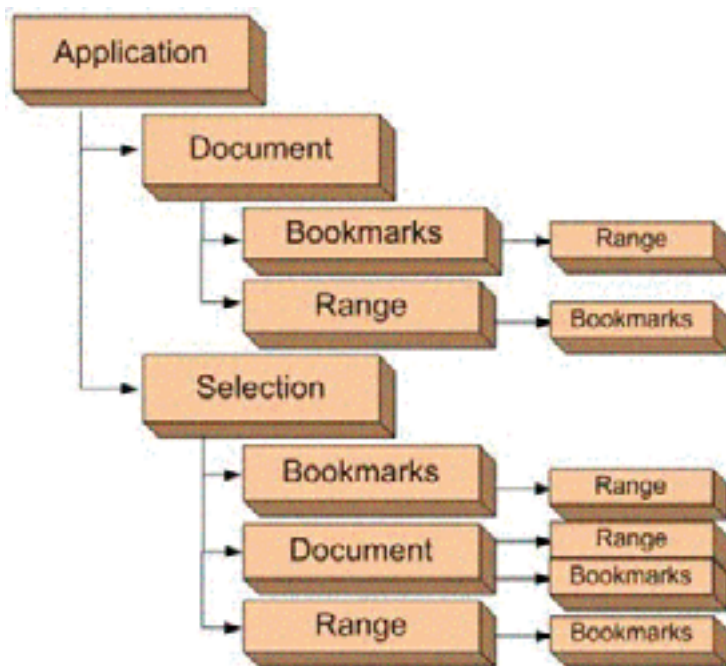


Schéma objektového modelu Word VSTO aplikácie

### Objekt Application

Objekt reprezentuje wordovú aplikáciu ako „rodičovskú“, zapuzdrujúcu všetky ostatné objekty. Metódy a ovládacie prvky sú rovnaké ako v prostredí programu Word.

### Objekt Document

Pre textový editor Word ani nemusíme zdôrazňovať, že jeho alfou a omegou je práca s dokumentmi. Podobne ako Word aj VSTO aplikácia vybudovaná nad týmto textovým procesorom môže dokumenty otvárať, zatvárať a vytvárať dokumenty nové. Viac dokumentov vnímame ako kolekciu, pričom jeden z nich je vždy aktívny.



## Objekt Selection

Ak chceme vo Worde pracovať s nejakou časťou textu, označíme ju najskôr ako blok a na tento blok následne aplikujeme príslušnú operáciu, napríklad vymazanie bloku, nahradenie textu v bloku iným textom, naformátovanie textu v bloku a podobne. Ak nie je vybraný žiadny blok, udejú sa príslušné zmeny vo vzťahu k vstupnému bodu. Podobne pracuje aj objekt Selection, ktorý môže byť tvorený aj viacerými nespojitými blokmi textu.

## Objekt Range

Objekt Range chápeme ako určitú spojitú oblasť dokumentu ohraničenú pozíciou prvého a posledného znaku. V dokumente môžeme podľa potreby týchto objektov definovať niekoľko

## Objekt Bookmark

Podobne ako objekt Range aj objekt Bookmark tvorí spojitú oblasť dokumentu ohraničenú pozíciou prvého a posledného znaku.

## Integrácia programu Office do vývojového prostredia

Skôr než predstavíme riešenia pre Word a Excel vo Visual Studio 2005 stojí za zmienku spôsob ako sa pre účely vývoja VSTO aplikácie integruje okno príslušného programu Office (Word Excel) do návrhového okna vývojového prostredia. Viac než rozsiahly slovný popis napovie obrázok. S Office dokumentom môžeme vo vývojovom prostredí normálne pracovať, pričom je možné využívať väčšinu funkcionality dostupnej v programoch tohto balíka.

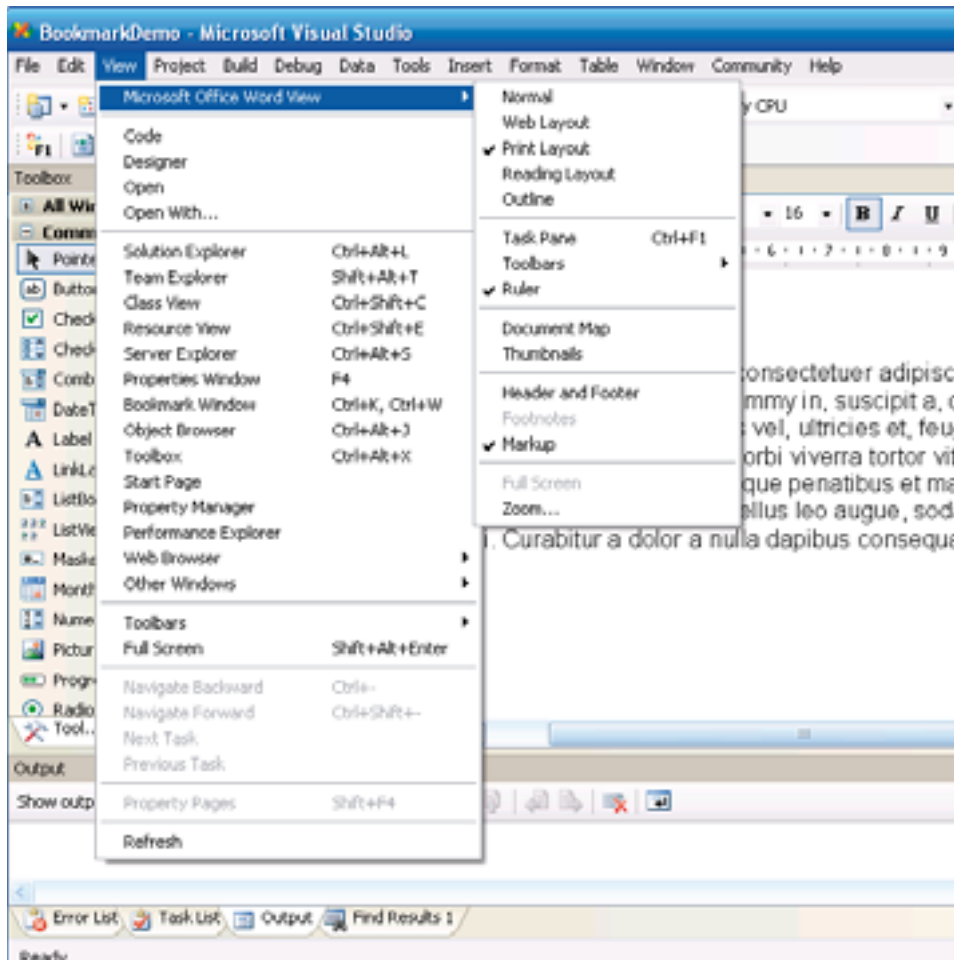


*Integrácia programu Office do vývojového prostredia*

Veľa funkcií programov kancelárskeho balíka Office (Word, Excel...) je v originálnych programoch dostupných pomocou menu a tlačidiel toolbaru. Pri integrácii takéhoto programu do vývojového prostredia bolo nutné integrovať aj systém menu. Napríklad do menu vývojového prostredia „View“ pribudla položka submenu „Microsoft Office Word View“, ktorá obsahuje položky menu „View“ pôvodnej aplikácie Office. Podobným spôsobom, napríklad cez kontextové menu sú dostupné aj mnohé



ďalšie funkcie. Ak si s niektorou funkciou nevieme rady, napríklad nevieme rýchlo nájsť ako zmeniť farbu fondu, naznačíme univerzálne riešenie. Stačí spustiť príslušný Office program, napríklad Word v novom okne a prekopírovať do neho blok textu, ktorý chceme netradičným spôsobom upraviť. Po vykonaných úpravách blok textu prekopírujeme naspäť do „wordového“ okna vývojového prostredia.



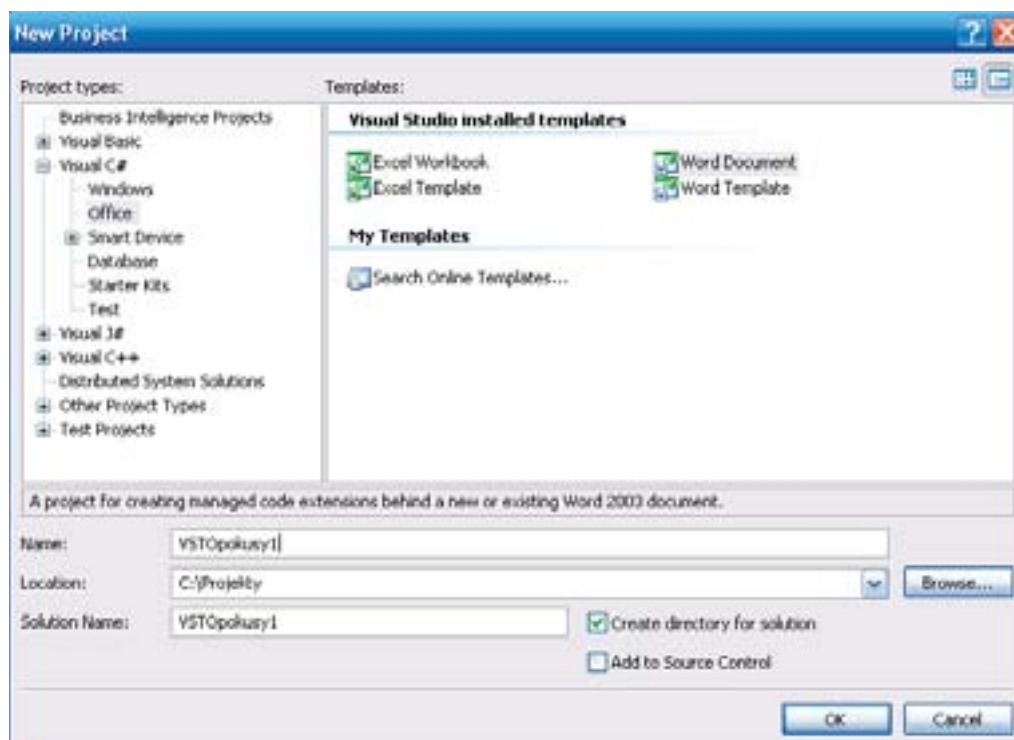
*Integrácia menu programu Office do menu vývojového prostredia*

## KAPITOLA 3: VYTVÁRANIE VSTO PROJEKTOV V EXCELI

Skôr, než sa pustíme do vývoja prvej aplikácie pre tabuľkový procesor Excel, prehladnime si zmeny, ktoré nastali vo vývojovom prostredí Visual Studio 2005 po nainštalovaní VSTO. V zložkách jednotlivých programovacích jazykov pribudli vnorené zložky Office, ktoré obsahujú šablóny „Office“ projektov.

- Excel Workbook
- Excel Template
- Word Document
- Word Template

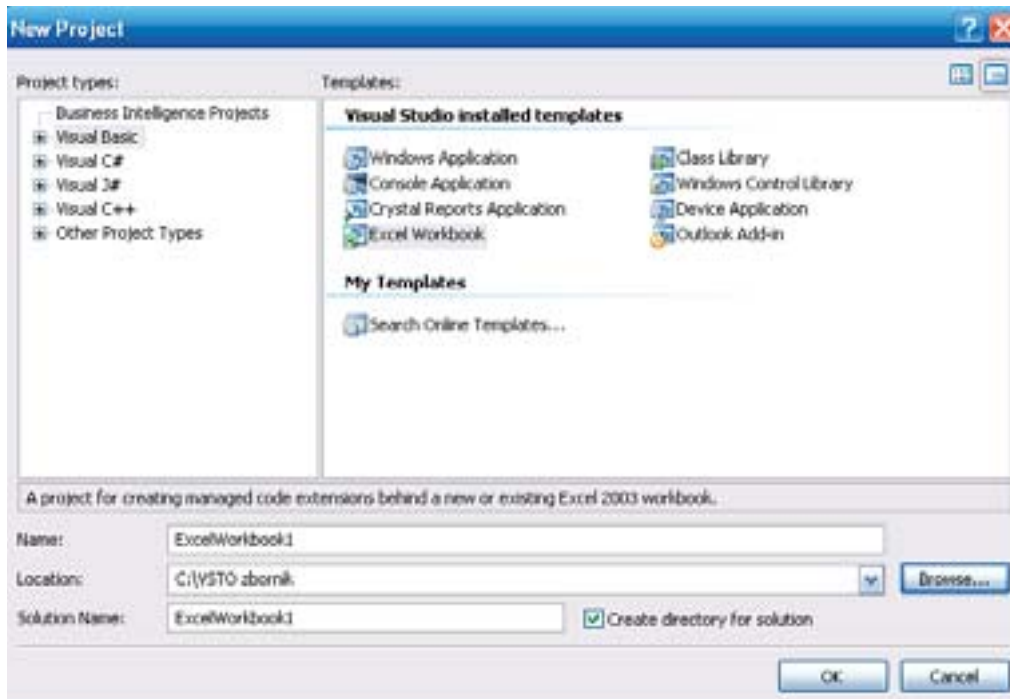
V prípade nainštalovania doplnku Outlook Add-in bude do zoznamu šablón doplnená aj šablóna projektu tohto typu. Podľa toho aký typ projektu potrebujeme a na ktorý programovací jazyk sa zameriame si vyberieme aj typ projektu. V tejto kapitole sa budeme venovať projektom pre Microsoft Excel a v nasledujúcej kapitole programom pre textový editor Microsoft Word.



Vytvorenie nového projektu pre Office v C#

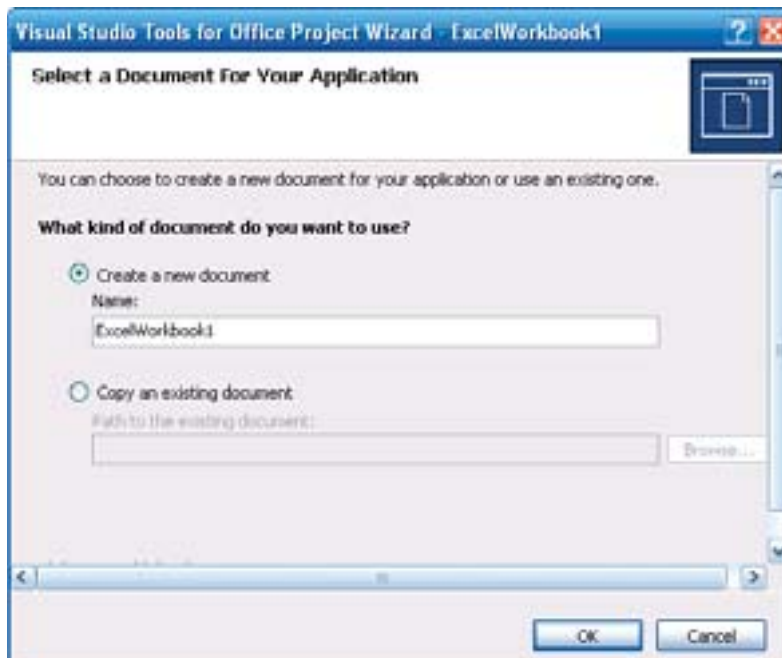
### Úvodný projekt typu „Hello World“

Začneme tradične – projektom typu „Hello World“ – teda výpisom textového reťazca na pracovnú plochu aplikácie. U aplikácií pre tabuľkový procesor Excel bude pracovnou plochou tabuľkový hárok. Vo vývojovom prostredí vytvoríme nový „Office“ projekt typu Excel Workbook. Podobne ako u iných typov projektov vyberieme lokalizáciu kam, do akého adresára budú súbory tvoriace projekt počas vývoja umiestnené. Podľa svojich skúseností vyberieme ako programovací jazyk buď Visual Basic, alebo C#.



*Vytvorenie nového VSTO projektu pre Excel vo VB 2005*

Prvým a vlastne jediným rozhodnutím špecifickým pre tento typ projektu v etape návrhu je, nad akým dokumentom programu Excel našu aplikáciu vytvoríme. Môže ísť o už existujúci dokument, alebo o nový dokument vytvorený v etape vytvárania VSTO projektu. V tejto prvej cvičnej aplikácii budeme pre jednoduchosť vychádzať z prázdneho, novovytvoreného dokumentu.



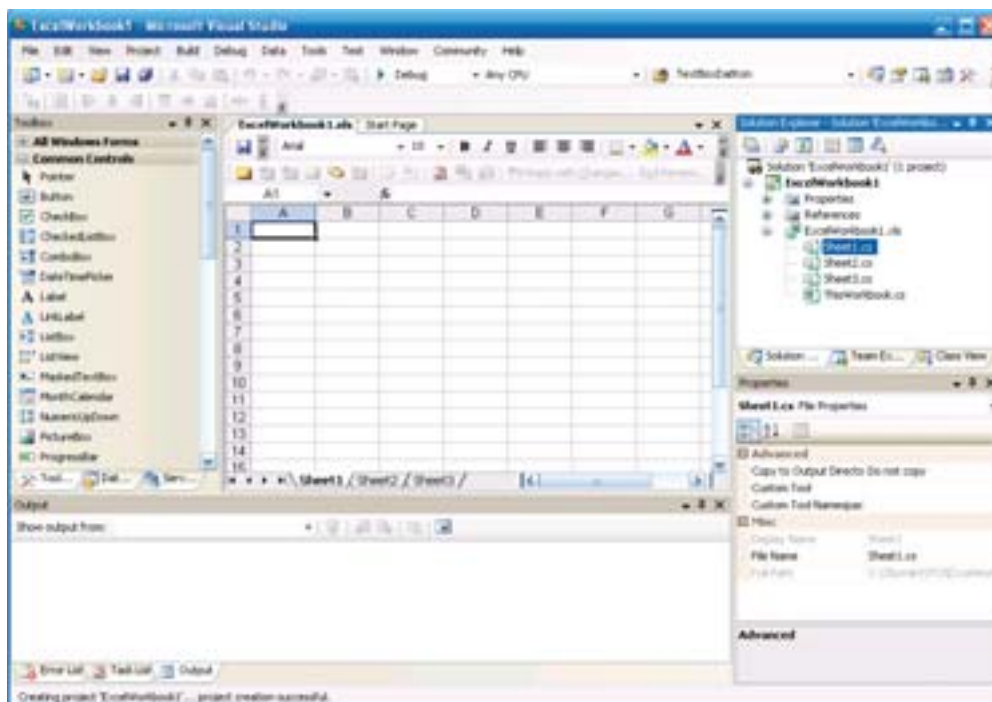
*Projekt môže pracovať s existujúcim dokumentom, prípadne je možné vytvoriť nový dokument*

Ak vytvárame svoj projekt ako prvý v novo nainštalovanom vývojovom prostredí, je potrebné explicitne povoliť prístup pre Visual Basic for Application project system.



*Povolenie prístupu pre Visual Basic for Application project system*

Pracovná obrazovka vývojového prostredia vo svojej centrálnej časti obsahuje plnohodnotné okno programu Excel. Môžeme v ňom vykonávať všetky úkony a operácie na tabuľkovom hárku, ktoré umožňuje Excel ako samostatný produkt, teda tak ako ho klasicky poznáme.



*Rozloženie pracovnej plochy vývojového prostredia pre projekt typu Excel Workbook*

Okrem prehliadky možností vizuálneho návrhu sa postupne zoznámime aj so zdrojovým kódom. Jadro „globálneho“ aplikačného kódu je v súbore ThisWorkbook.cs. Po vytvorení nového projektu sú v parciálnej triede ThisWorkbook len dve metódy. Jedna z nich je aktivovaná pri štarte a druhá pri ukončení aplikácie.

## Visual Basic

*Public Class ThisWorkbook*

*Private Sub ThisWorkbook\_Startup(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Startup  
End Sub*

*Private Sub ThisWorkbook\_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Shutdown  
End Sub*

*End Class*

## C#

```
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.VisualStudio.Tools.Applications.Runtime;
using Excel = Microsoft.Office.Interop.Excel;
using Office = Microsoft.Office.Core;

namespace ExcelWorkbook1c
{
    public partial class ThisWorkbook
    {
        private void ThisWorkbook_Startup(object sender, System.EventArgs e)
        {
        }

        private void ThisWorkbook_Shutdown(object sender, System.EventArgs e)
        {
        }
    }
}
```

Z najvyššej úrovne excelovského dokumentu sa môžeme presunúť na nižšiu úroveň hierarchie, na úroveň kódu jednotlivých hárkov.

## Visual Basic

```
Public Class Harok1
    Private Sub Harok1_Startup(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Startup

    End Sub

    Private Sub Harok1_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Shutdown

    End Sub
End Class
```

## C#

```
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.VisualStudio.Tools.Applications.Runtime;
using Excel = Microsoft.Office.Interop.Excel;
using Office = Microsoft.Office.Core;

namespace ExcelWorkbook1c
{
    public partial class Hárok1
    {
        private void Harok1_Startup(object sender, System.EventArgs e)
        {
        }
    }
}
```

```

private void Harok1_Shutdown(object sender, System.EventArgs e)
{
}
}
}

```

Aby sme neporušili tradíciu, že prvá aplikácia má byť typu „Helo World“, zostáva nám vypísať text do niektorej bunky Excelovskej tabuľky. Kód pre výpis textu vložíme do metódy Hárók1\_Startup.

### Visual Basic

```

Private Sub Hárók1_Startup(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Startup
    Dim rng As Excel.Range = Globals.Harok1.Range(„A1“)
    rng.Value2 = „Nazdar svet“
End Sub

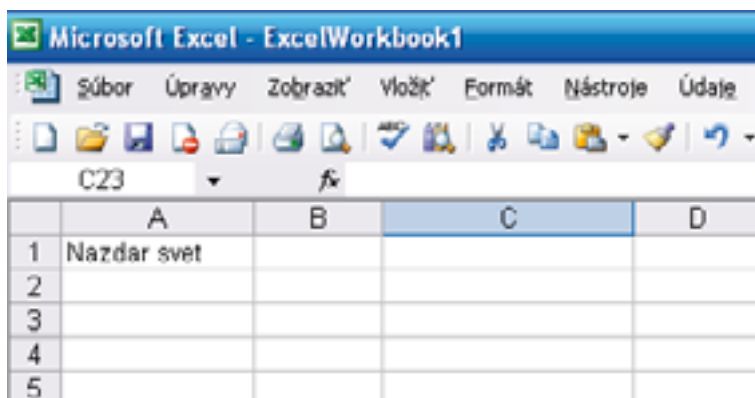
```

### C#

```

private void Harok1_Startup(object sender, System.EventArgs e)
{
    Excel.Range rng = Globals.Harok1.Range[„A1“, missing];
    rng.Value2 = „Nazdar svet“;
}

```

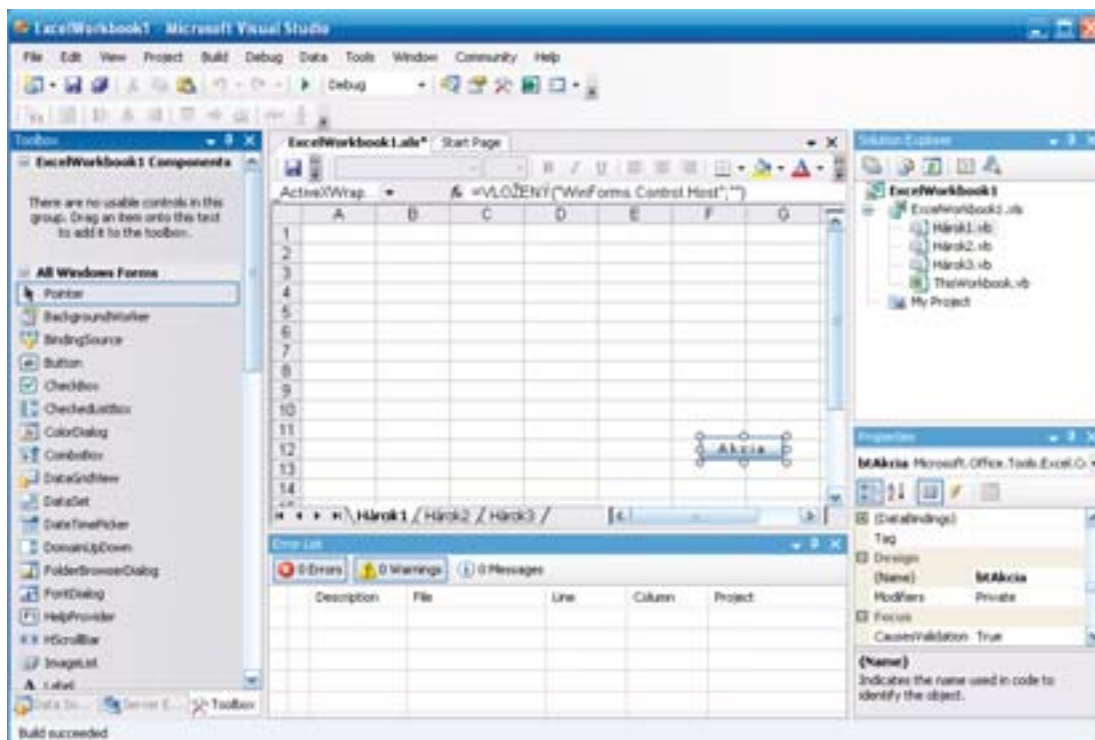


VSTO aplikácia typu „Hello World“ pre MS Excel

## Práca s dokumentom MS Excel

Pri rozbore možností práce s dokumentmi programu Office sme problematiku rozčlenili na dve oblasti. Jednak manipuláciu s dokumentom ako celkom a jednak práca s obsahom dokumentu. Po tomto základnom filozofickom rozdelení sa ponúkajú ďalšie rozdelenia, tentoraz implementačné. Na vyššom stupni hierarchie je rozdelenie na dokumenty programu Excel a Word. Na nižšom stupni hierarchie môžeme akceptovať rozdelenie podľa použitého programovacieho jazyka – C# alebo Visual Basic.

Aby sme nemuseli zakaždým uvádzať kód tela procedúry, ale len krátky stručný a prehľadný kód, vytvoríme si testovaciu zostavu pre experimenty s obsahom dokumentu. Vytvoríme projekt v príslušnom programovacom jazyku a na plochu prvého pracovného hárku Excelu umiestnime tlačidlo.



Tlačidlo pre testovanie čiastkových akcií pri manipulácii s dokumentom

Všimnite si v okne Solution Explorer hierarchiu zdrojových kódov. Na najvyššej úrovni je Workbook a teda v našom konkrétnom príklade kód ThisWorkbook.vb. Na nižšej úrovni hierarchie sú kódy jednotlivých riadkov (Hárak1.vb, Hárak2.vb, Hárak3.vb,)

Dvojklikom na symbol tlačidla vytvoríme telo obslužnej procedúry.

### Visual Basic

```
Private Sub btAkcia_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btAkcia.Click
```

```
End Sub
```

### C#

```
private void btAkcia_Click(object sender, EventArgs e)
{
}
}
```



Je potrebné si uvedomiť, že obslužná procedúra sa vzťahuje k prvému hároku. Takýmto spôsobom dokážeme pracovať s jeho obsahom. Ak chceme pomocou prvkov umiestnených na ploche lokálnych hárkov pracovať na úrovni globálneho kódu, teda kódu v súbore „ThisWorkbook.vb“ môžeme použiť jeden malý trik.

V súbore ThisWorkbook.vb vytvoríme procedúry pre jednotlivé akcie do ktorých umiestnime výkonný kód a tieto procedúry budeme cez namespace Globals volať z kódov jednotlivých hárkov. V praxi to môže byť realizované takto

### Visual Basic

ThisWorkbook.vb

```
Public Sub AkciaWorkbooku()  
    '... vykonny kod  
End Sub
```

Hárok1.vb

```
Private Sub btAkcia_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btAkcia.Click  
    Globals.ThisWorkbook.AkciaWorkbooku()  
  
End Sub
```

### C#

ThisWorkbook.cs

```
public void AkciaWorkbooku()  
{  
    //... vykonny kod  
}
```

Hárok1.cs

```
private void btAkcia_Click(object sender, EventArgs e)  
{  
    Globals.ThisWorkbook.AkciaWorkbooku()  
}
```

## Excel - práca s obsahom dokumentu

Aj napriek tomu, že táto publikácia je koncipovaná prakticky, je potrebné si vždy aspoň zbežne predstaviť základné objekty a ich hierarchiu. U Excelu môžeme túto hierarchiu znázorniť schematicky takto:

**Application**  
**Workbook**  
**Worksheet**  
**Range**

## Objekt Application

Objekt reprezentuje wordovú aplikáciu ako „rodičovskú“, zapuzdrujúcu všetky ostatné objekty. Metódy a ovládacie prvky sú rovnaké ako v prostredí programu Excel.

## Objekt Workbook

Dokument programu Excel je rozčlenený na jednotlivé zošity (workbooks). Pomocou aplikačného kódu môžeme jednotlivé zošity vytvárať, otvárať a zatvárať. Zostavu viacerých zošitov vnímame ako kolekciu, pričom jeden z nich je vždy aktívny.

## Objekt Selection

Ak chceme vo Worde pracovať s nejakou časťou textu, označíme ju najskôr ako blok a na tento blok následne aplikujeme príslušnú operáciu, napríklad vymazanie bloku, nahradenie textu v bloku iným textom, naformátovanie textu v bloku a podobne. Ak nie je vybraný žiadny blok, udejú sa príslušné zmeny vo vzťahu k vstupnému bodu. Objekt Selection môže byť tvorený aj viacerými nespojitými blokmi textu.

## Objekt Range

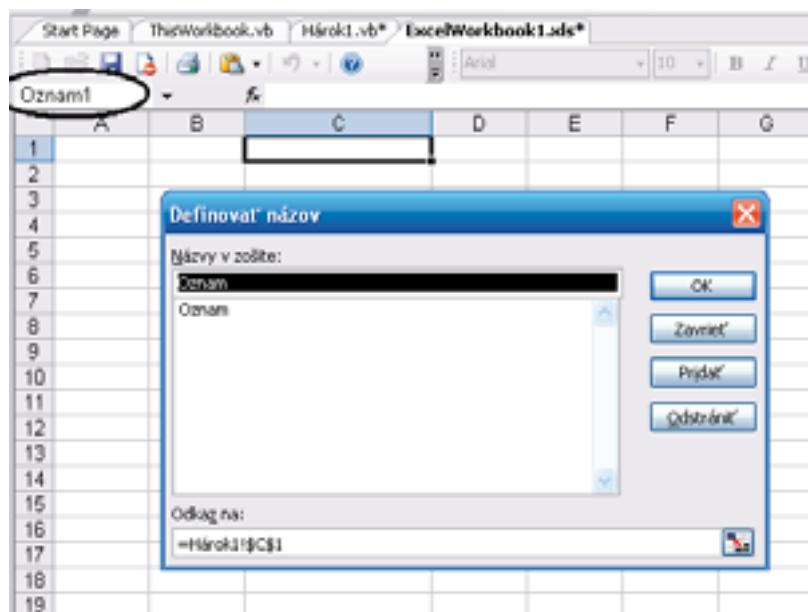
Objekt Range chápeme ako určitú spojitú oblasť dokumentu ohraničenú pozíciou prvého a posledného znaku. V dokumente môžeme podľa potreby týchto objektov definovať niekoľko.

## Objekt Bookmark

Podobne ako objekt Range aj objekt Bookmark tvorí spojitú oblasť dokumentu ohraničenú pozíciou prvého a posledného znaku.

## Práca s tabuľkami

Keďže program Excel je vo svojej podstate tabuľkový procesor, alfou a omegou práce v tomto programe, či už manuálnej, alebo automatizovanej bude práca s tabuľkami, s ich riadkami, stĺpcami a bunkami. Všimnite si, že bunky tabuľky sú adresované v osi x (po stĺpcoch) písmenami abecedy, teda A, B, C, D, E,... a v osi y (po riadkoch), číslami, teda 1, 2, 3, 4, 5,.. Ľavá horná bunka tabuľky má v tomto systéme adresu A1, bunka vpravo od nej B1, bunka pod ňou A2. Bunky, alebo rozsahy buniek však môžeme aj pomenovať. V našom prípade sme premenovali bunku C1 na „Adresa“.



Premenovanie bunky

Bunku je možné premenovať buď pomocou Dialógu aktivovanom kontextovým menu „Manage Names Range“ alebo pomocou editačného poľa v ľavej časti záhlavia hárku.

### Visual Basic

```
Me.Oznam.Value = „Nazdar“
```

### C#

```
this.Oznam.Value2 = „Nazdar“;
```

Bunku tabuľky môžeme adresovať nielen cez jeho pomenovanie ale aj priamo. Napríklad bunka vľavo hore má súradnice A1.

### Visual Basic

```
Dim rng As Excel.Range = Globals.Hárok1.Range(„A1“)  
rng.Value2 = „Nazdar“
```

### C#

```
Excel.Range rng = Globals.Hárok1.Range[„A1“, missing];  
rng.Value2 = „Nazdar“;
```

Bunku alebo množinu buniek môžeme pomenovať aj programovo napríklad

### Visual Basic

```
Dim NamedRange1 As Microsoft.Office.Tools.Excel.NamedRange = _  
    Globals.Hárok1.Controls.AddNamedRange(_  
        Globals.Hárok1.Range(„A5“), „Bunka5“)  
Me.Range(„Bunka5“).Value = „Nazdar“
```

### C#

```
Microsoft.Office.Tools.Excel.NamedRange NamedRange1 =  
    Globals.Hárok1.Controls.AddNamedRange(  
        Globals.Hárok1.Range[„A5“, missing], „Bunka5“);
```

```
NamedRange1.Offset[index, 0].Value2 = „Nazdar“;;
```

## Použitie funkcií

Prakticky každý kto pracoval s programom Excel využil minimálne niekoľko z veľmi širokého spektra rôznych funkcií, či už matematických, štatistických. Aby sme ukázali použitie funkcií vyrobíme jednoduchý príklad, ktorý do prvého stĺpca vypíše 20 náhodne vygenerovaných čísel a následne na ne aplikuje funkcie Max, Min, a Average (aritmetický priemer). Výsledky vypíšeme do buniek vpravo od stĺpca čísel.

### Visual Basic

```
Dim rnd As New System.Random  
Dim rng As Excel.Range = Globals.Hárok1.Range(„A1“, „A20“)  
Dim i As Integer  
For i = 1 To 20  
Me.Cells(i, 1) = rnd.Next(100)  
Next i
```

Me.Cells(2, 3) = „Min“  
 Me.Cells(2, 4) = Me.Application.WorksheetFunction.Min(rng)  
 Me.Cells(3, 3) = „Max“  
 Me.Cells(3, 4) = Me.Application.WorksheetFunction.Max(rng)  
 Me.Cells(4, 3) = „Average“  
 Me.Cells(4, 4) = Me.Application.WorksheetFunction.Average(rng)

	A	B	C	D	E
1	12				
2	11		Min	5	
3	5		Max	98	
4	52		Average	49,2	
5	71				
6	37				
7	21				
8	75				
9	12				
10	78		Akcia		
11	63				
12	13				
13	9				
14	63				
15	98				
16	65				
17	20				
18	94				
19	91				
20	94				

Príklad pre využitie funkcií

Takmer nikde sme sa nepristavovali na rozdieloch medzi Visual Basicom a C#, dokonca sme v úvode odporučili C# ako plnohodnotný programovací jazyk pre VSTO aplikácie. Má to však jeden háčik a to sú chýbajúce parametre funkcií. Funkcie ich majú definovaných 30, z nich však využijeme len niektoré, minimálne však jeden. Pre ostatné, voliteľné parametre odovzdáme referenciu na objekt Nic (znamená to doslova a do písmena nič), čiže anglicky Missing Value. Namiesto vytvárania objektu „chýbajúcej hodnoty“ môžeme použiť referenciu na objekt „missing“. Dôsledkom uvedenej skutočnosti bude kód v jazyku C# oveľa rozsiahlejší. V našom výpise je pre úsporu miesta len kód pre funkciu „Min“.

## C#

```

System.Random rnd = new System.Random();
Excel.Range rng = Globals.Hárok1.Range[„A1“, „A20“];
object Nic = System.Reflection.Missing.Value;
for ( int i = 1 ; i <= 20; i++)
    this.Cells[i, 1] = rnd.Next(100);

this.Cells[2, 3] = „Min“;
this.Cells[2, 4] = this.Application.WorksheetFunction.Min(rng,
    Nic, Nic, Nic, Nic, Nic, Nic, Nic, Nic,
    Nic, Nic, Nic, Nic, Nic, Nic, Nic, Nic,
    Nic, Nic, Nic, Nic, Nic, Nic, Nic, Nic,
    Nic, Nic, Nic, Nic, Nic);
  
```

## Utriedenie údajov

Pri pohľade na stĺpec náhodne vygenerovaných čísel z príkladu v predchádzajúcej state mnohých určite napadne myšlienka utriediť ich. Aj v tomto prípade je nevýhodou jazyka C# nutnosť vymenovania všetkých nepovinných (optional) parametrov.

### Visual Basic

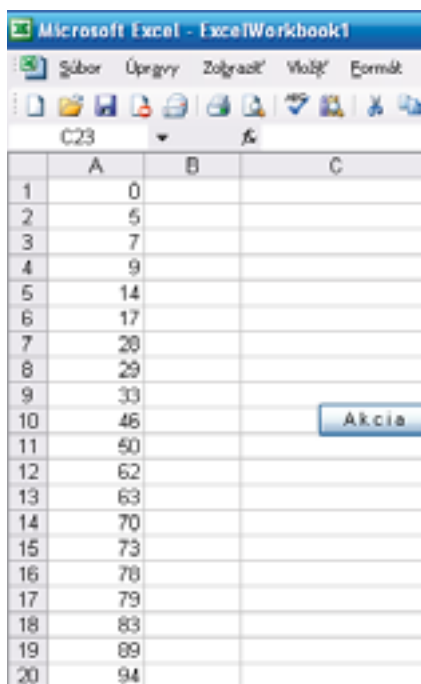
```
Dim rnd As New System.Random
Dim rng As Excel.Range = Globals.Hárok1.Range(„A1“, „A20“)
Dim i As Integer
For i = 1 To 20
    Me.Cells(i, 1) = rnd.Next(100)
Next i

rng.Sort(rng, Orientation:=Excel.XlSortOrientation.xlSortColumns)
```

### C#

```
System.Random rnd = new System.Random();
Excel.Range rng = Globals.Hárok1.Range[„A1“, „A20“];
object Nic = System.Reflection.Missing.Value;
for ( int i = 1 ; i <= 20; i++)
    this.Cells[i, 1] = rnd.Next(100);

rng.Sort(rng, Excel.XlSortOrder.xlAscending,
    Nic, Nic, Excel.XlSortOrder.xlAscending,
    Nic, Excel.XlSortOrder.xlAscending,
    Excel.XlYesNoGuess.xlNo, Nic, Nic,
    Excel.XlSortOrientation.xlSortColumns,
    Excel.XlSortMethod.xlPinYin,
    Excel.XlSortDataOption.xlSortNormal,
    Excel.XlSortDataOption.xlSortNormal,
    Excel.XlSortDataOption.xlSortNormal);
```



	A	B	C
1	0		
2	5		
3	7		
4	9		
5	14		
6	17		
7	28		
8	29		
9	33		
10	46		Akcia
11	50		
12	62		
13	63		
14	70		
15	73		
16	78		
17	79		
18	83		
19	89		
20	94		

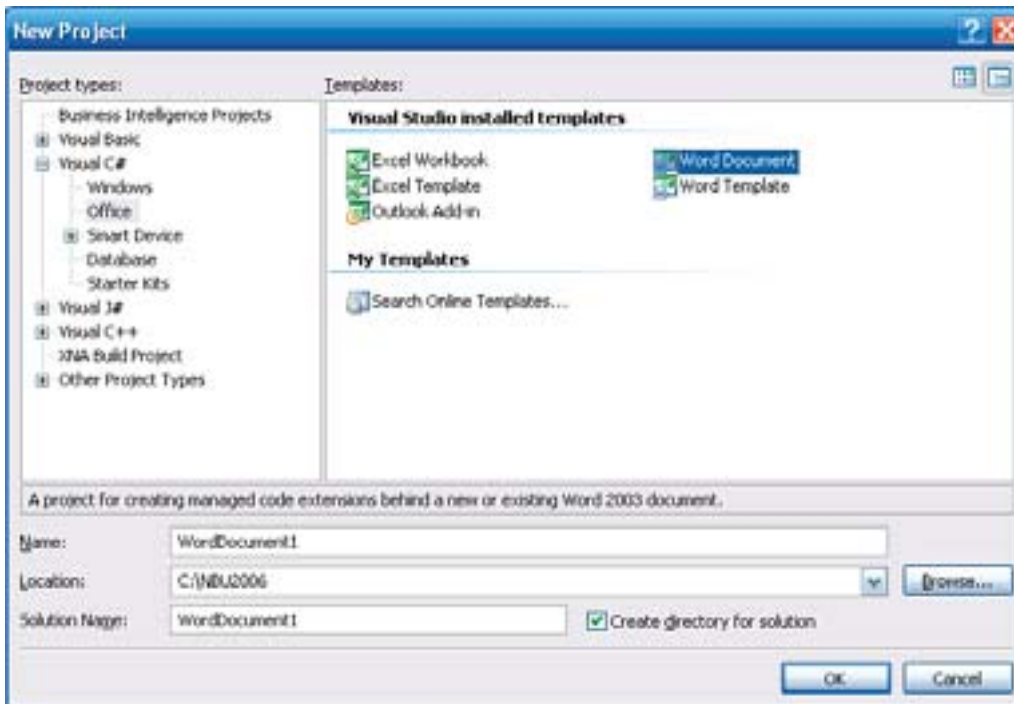
Utriedenie údajov

## KAPITOLA 4: VYTVORENIE VSTO PROJEKTOV VO WORDE

Zatiaľ čo aplikácie pre Excel sa primárne najviac hodia pre výstup dokumentov obsahujúcich tabuľky, pre aplikácie generujúce všeobecné dokumenty je výhodnejšie využiť VSTO aplikáciu založenú na textovom editore Microsoft Word.

Projekty VSTO aplikácií pre program Word sa vytvárajú rovnakým postupom ako programy pre Excel. V menu vytvoríme nový „Office“ projekt typu Word. Na výber máme dve možnosti.

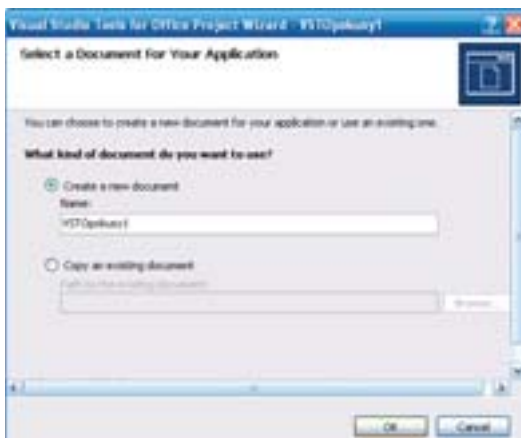
- Word Document
- Word Template



*Dialóg pre vytvorenie nového VSTO projektu pre MS Word*

Pre jednoduché cvičné projekty z tohto zborníka budeme vytvárať projekty typu Word Document.

V procese vytvárania novej aplikácie je len jediné rázcestie, kedy špecifikujeme, či budeme pracovať s novým, alebo s už existujúcim dokumentom. Presnejšie, ak si vyberieme už existujúci dokument, vytvorí sa jeho kópia v adresári projektu, to znamená, že „zdrojový“ dokument zostane mimo projekt. Pre náš pokusný projekt môžeme otvoriť ľubovoľný dokument obsahujúci nejaký text, prípadne vytvoriť nový dokument a do neho prekopírovať niekoľko odstavcov textu z iného dokumentu.



*Projekt môže pracovať s existujúcim dokumentom, prípadne je možné vytvoriť nový dokument.*

Námetom prvého cvičného príkladu by mal byť tradične výpis cvičného textu „Hello word“. Pre tento typ projektu je vlastne výpis textu a to v rôznych podobách, formátoch a úpravách dokonca primárnou úlohou, takže spôsob kam umiestniť text a ako ho zobrazíť si môžeme vybrať z viacerých možností. Aby sme si zvykli na typickú situáciu, kedy VSTO aplikácia pracuje s existujúcim textom, budeme v niektorých príkladoch ako východiskovú situáciu využívať dokument obsahujúci časť štandardného medzinárodného typografického textu „Lorem Ipsum“. Ako uvidíme neskôr, pre niektoré príklady je potrebné aby bol tento cvičný text dlhší a mal aspoň tri odstavce.

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed suscipit sollicitudin nunc. Etiam lectus felis, nonummy in, suscipit a, condimentum vitae, mi. Nunc est. Praesent ipsum sem, sodales vel, ultricies et, feugiat vitae, diam. Vestibulum eros. Duis ornare lorem id ligula. Morbi viverra tortor vitae metus. Nulla velit. Cras hendrerit vehicula leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Phasellus leo augue, sodales vitae, interdum sit amet, feugiat at, mi. Curabitur a dolor a nulla dapibus consequat.*

*Aenean mattis egestas pede. Sed interdum sollicitudin magna. Donec laoreet, nunc id malesuada ullamcorper, lacus augue pharetra dui, eget pulvinar risus dolor sit amet turpis. Vestibulum suscipit, dolor tristique commodo sodales, massa mauris ultricies orci, non convallis dui odio vitae tortor. Ut ultrices nulla at arcu venenatis gravida. Quisque tempus ante eget nisi. Duis consectetur mauris quis urna. Aliquam pharetra, sapien sed convallis dapibus, nisl enim facilisis nisl, ut feugiat quam mi non sem. Vivamus dictum nunc a risus. Mauris ipsum. Vestibulum leo. Praesent ac est. Sed porttitor augue ut augue. Phasellus aliquam, justo varius commodo tempor, erat magna laoreet quam, sagittis tristique lorem diam vitae nibh.*

*Integer ante purus, egestas eget, feugiat vel, sagittis imperdiet, elit. Duis fringilla blandit pede. Curabitur diam orci, iaculis ut, molestie a, rhoncus sed, augue. Duis porta nisi ut diam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In tempus urna sit amet magna. Vestibulum lectus sapien, lacinia a, facilisis venenatis, gravida et, erat. Donec pellentesque elit sit amet ante. Integer sodales. Curabitur feugiat metus et nulla. Pellentesque ac leo eget sapien lobortis vehicula. Fusce luctus est vitae odio. In hac habitasse platea dictumst.*

## Základy práce s ovládacími prvky

Prvou pravdepodobne najjednoduchšou úlohou bude zmena textu vo vnútri ovládacieho prvku. Východiskovou situáciou bude prázdny dokument. Priamo na plochu wordového dokumentu umiestnime dva ovládacie prvky – Textbox a Button.

### Textbox

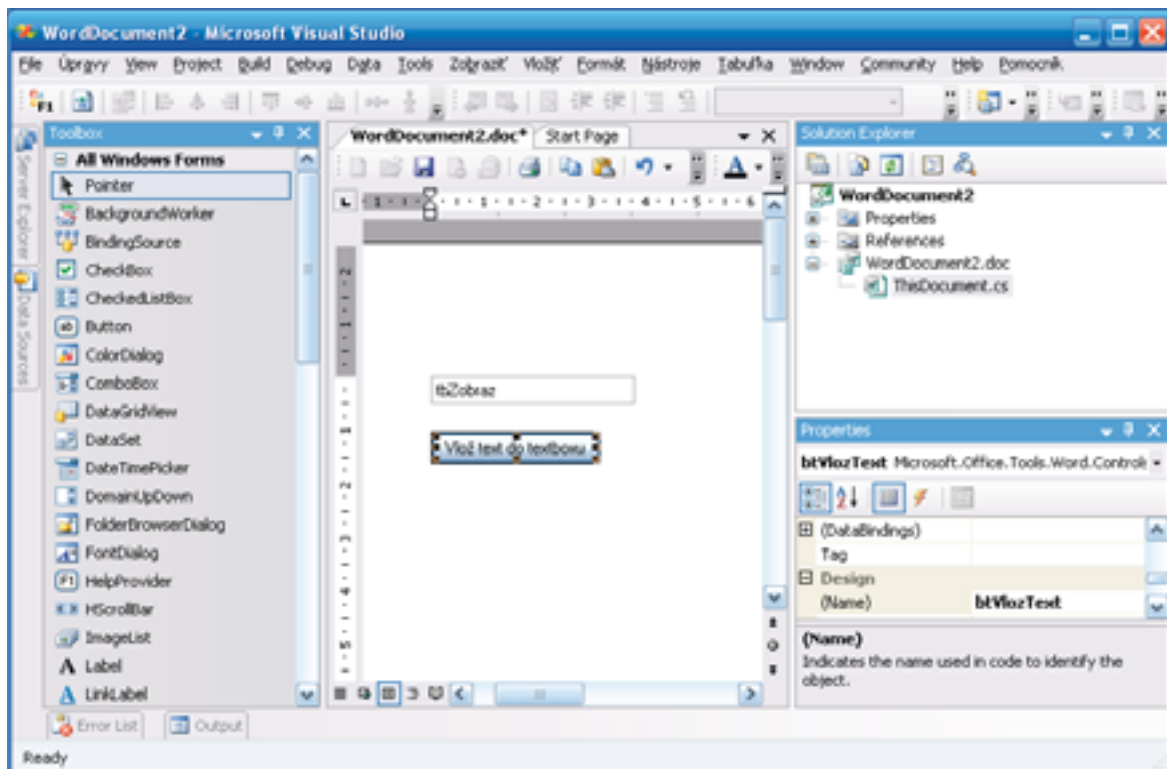
Parameter	Hodnota
Name	tbZobraz

### Button

Parameter	Hodnota
Name	btVlozText
Text	Vlož do textboxu

Prvky vytvoríme presunom symbolov „textbox“ a „button“ z toolboxu na plochu dokumentu.





Návrhové zobrazenie aplikácie

Na ploche aplikácie je zatiaľ jeden textbox a jedno tlačidlo. Ak by sme teraz aplikáciu preložili a spustili fungovala by síce, no robila by presne to, čo sme zatiaľ naprogramovali – teda nič, len by zobrazila spomínané dva prvky. Aby sme splnili zadanie, je potrebné do aplikácie doplniť kód pre obsluhu zatlačenia tlačidla.

Dvojklikom na tlačidlo vytvoríme procedúru pre obsluhu udalosti jeho zatlačenia.

```
private void btVlozText_Click(object sender, EventArgs e)
{
    this.tbZobraz.Text = „Hello World!";
}
```

### Poznámka

V tele tejto procedúry môžeme bez toho aby sme museli vytvárať nové projekty postupne skúšať preberané úkony a princípy z ďalších príkladov tohto zborníka, napríklad označovanie blokov textu. Vtedy samozrejme do dokumentu prekopírujeme cez clipboard cvičný text.

Kompletný zdrojový kód v súbore ThisDocument.cs potom bude

```
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.VisualStudio.Tools.Applications.Runtime;
using Word = Microsoft.Office.Interop.Word;
using Office = Microsoft.Office.Core;

namespace WordDocument2
{
    public partial class ThisDocument
    {
```

```
private void ThisDocument_Startup(object sender, System.EventArgs e)
{
}
```

```
private void ThisDocument_Shutdown(object sender, System.EventArgs e)
{
}
```

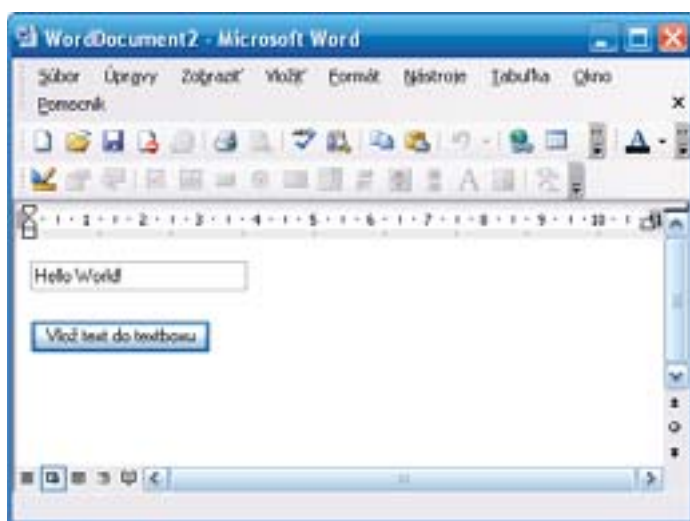
*VSTO Designer generated code*

```
private void btVlozText_Click(object sender, EventArgs e)
{
    this.tbZobraz.Text = „Hello World!“;
}
}
```

Pre úplnosť si všimnite si, že časť kódu sa skrýva za odkazom *VSTO Designer generated code*. Je to kód, ktorý vznikol v etape vizuálneho návrhu.

```
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InternalStartup()
{
    this.btVlozText.Click += new
    System.EventHandler(this.btVlozText_Click);
    this.Startup += new System.EventHandler(this.ThisDocument_Startup);
    this.Shutdown += new System.EventHandler(this.ThisDocument_Shutdown);
}
}
```

Teraz môžeme aplikáciu preložiť a spustiť. Po zatlačení tlačidla by sa mal v editačnom okienku textboxu objaviť príslušný text.



*Test funkcie VSTO aplikácie pre výpis textu pomocou ovládacieho prvku*

## Vloženie textu do dokumentu textového editora Word

Predchádzajúci príklad bol síce jednoduchý a názorný, no z hľadiska aplikačnej logiky dosť umelý. Ak máme k dispozícii zobrazovacie možnosti aplikácie Word, prečo by sme sa mali držať zvyklostí z klasických WinForm aplikácií a text vypisovať pomocou ovládacích prvkov, keď ho môžeme vložiť priamo do dokumentu. Pre tento účel môžeme vytvoriť nový projekt s jedným jediným tlačidlom, alebo jednoduchšia a rýchlejšia varianta bude pridať do projektu z predchádzajúcej state nové tlačidlo a vymazať z neho nepotrebný textox.

### Button

Parameter	Hodnota
Name	btVlozText
Text	Vlož do textboxu

Kód pre obsluhu tlačidla bude

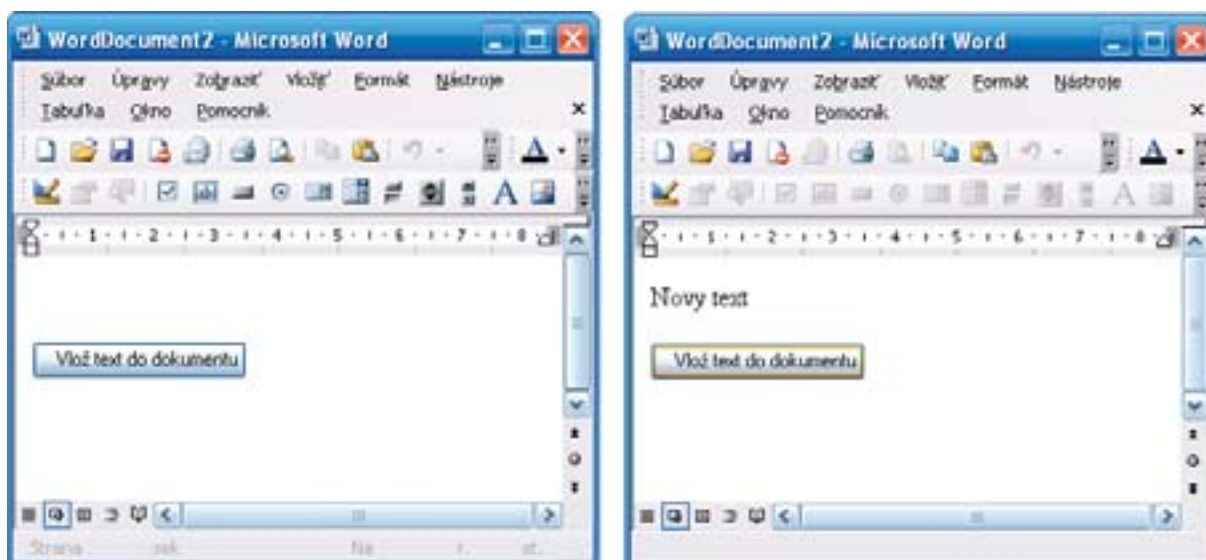
```
private void btVlozText_Click(object sender, EventArgs e)
{
    object start = 0;
    object end = 0;

    Word.Range rng = this.Range(ref start, ref end);
    rng.Text = „Novy text „;
}
```

### Visual Basic

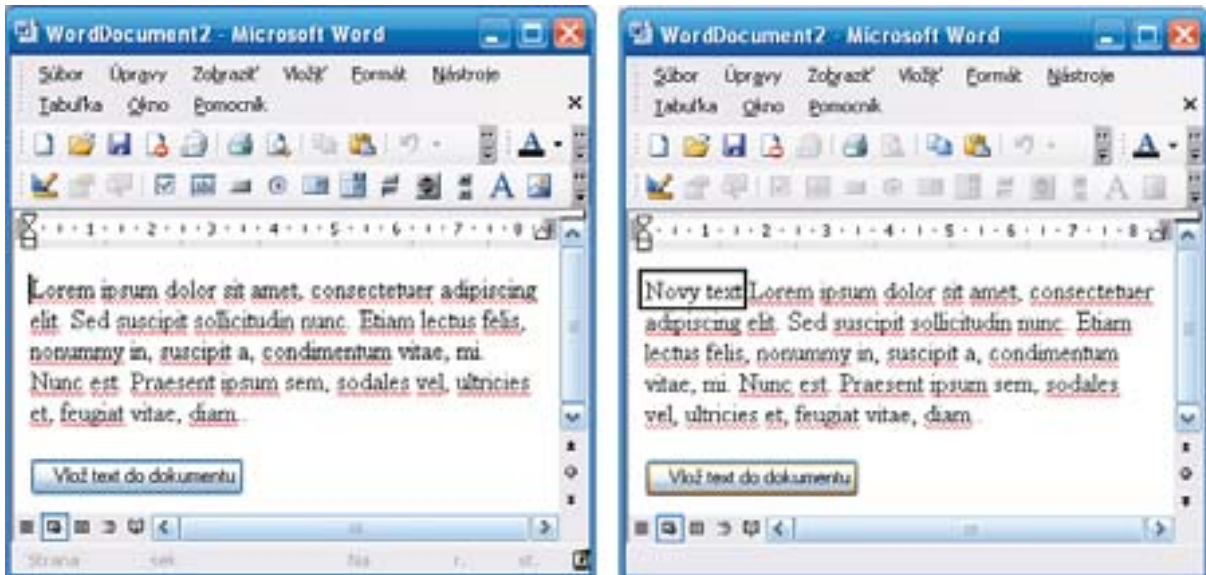
```
Dim rng As Word.Range = Me.Range(Start:=0, End:=7)
rng.Select()
```

Pomocou objektu Range špecifikujeme, kam chceme text vložiť. U prázdneho dokumentu to bude na pozíciu 0.



Test funkcie VSTO aplikácie pre výpis textu do dokumentu. Vľavo situácia pred a vpravo po zatlačení tlačidla

Ten istý príklad môžeme vyskúšať aj v inej variante. Východisková situácia nebude prázdny dokument, ale do dokumentu zobrazenom v okne vývojového prostredia Visual Studio 2005 vložíme na začiatok dokumentu nad tlačidlo nejaký text, čiže inými slovami nebudeme pracovať s prázdny, ale so všeobecným dokumentom.



Test funkcie VSTO aplikácie pre výpis textu do dokumentu. Vľavo situácia pred a vpravo po zatlačení tlačidla. Pridaný text je zvýraznený rámkom

A do tretice môžeme vyskúšať variantu, kedy spustíme aplikáciu s prázdny dokumentom, obsahujúci len samotné tlačidlo. Po spustení aplikácie do jej dokumentu skopírujeme text a až následne aktivujeme tlačidlo pre výpis textu. Konečný výsledok je rovnaký ako v predchádzajúcom príklade.

## Výber celého dokumentu

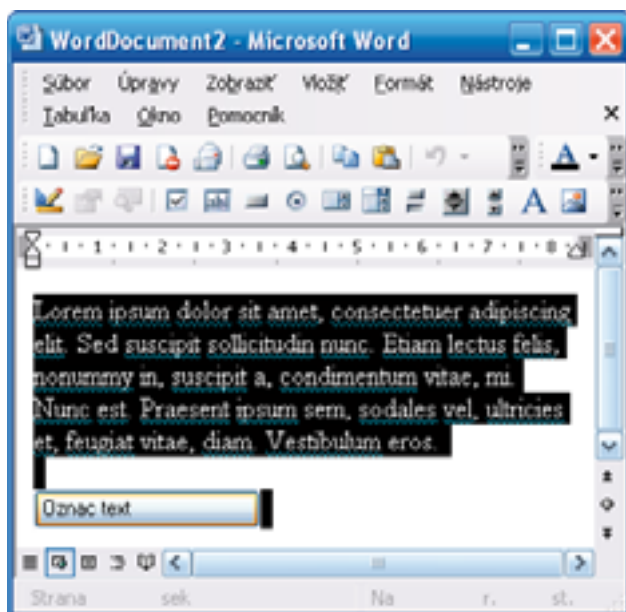
Po prvých úspešných pokusoch s výpisom začnú naše nároky a požiadavky rásť. Začnime označovaním blokov textu, nakoľko práve takto pomocou blokov sa pracuje často aj priamo v aplikácii Word. Najskôr označme celý text.

### C#

```
private void btVyberText_Click(object sender, EventArgs e)
{
    object start = this.Content.Start;
    object end = this.Content.End;
    this.Range(ref start, ref end).Select();
}
```

### Visual Basic

```
Me.Range.Select()
```



Výber celého dokumentu

Táto funkcia môže byť veľmi užitočná, napríklad ak pracujeme z viacerými dokumentami. Ak otvoríme okrem aktívneho dokumentu ešte ďalší, tento sa otvorí v ďalšom okne, pričom kód VSTO aplikácie dokáže pracovať s inými dokumentami. Nie vždy však bude používateľ ochotný prepínať sa medzi oknami, uvítal by keby aj druhý dokument bol otvorený v aktívnom okne. Vtedy si môžeme pomôcť malým trikom, kedy v oboch oknách označíme celý text ako blok a tieto bloky vzájomne vymeníme, alebo len prekopírujeme blok textu z okna na pozadí do aktívneho okna. Preto trochu predbehneme výklad a uvedieme fragment kódu pre výmenu obsahu medzi dvoma dokumentami.

```
public void VymenDokument()
{
    //Vymena blokov medzi dvoma dokumentami
    object index0 = 1;
    Word.Document doc0 = Application.Documents.get_Item(ref index0);
    object start0 = doc0.Content.Start;
    object end0 = doc0.Content.End;
    Word.Range rng0 = doc0.Range(ref start0, ref end0);
    rng0.Select();

    object index1 = 2;
    Word.Document doc1 = Application.Documents.get_Item(ref index1);
    object start1 = doc1.Content.Start;
    object end1 = doc1.Content.End;
    Word.Range rng1 = doc1.Range(ref start1, ref end1);
    rng1.Select();
    rng0.Copy();
    rng1.Paste();
}
```

## Výber časti dokumentu

Pri práci s jedným dokumentom má význam označiť ako blok určitú časť textu, napríklad vetu, odsek, prípadne explicitne stanovený úsek textu. Začneme kódom pre označenie explicitne stanoveného bloku. Napríklad ak potrebujeme označiť text od stého po sto desiaty znak.

```
private void btVyberBlok_Click(object sender, EventArgs e)
{
    object start = 100;
    object end = 110;
    Word.Range rng = this.Range(ref start, ref end);
    rng.Select();
}
```

## Výber vety

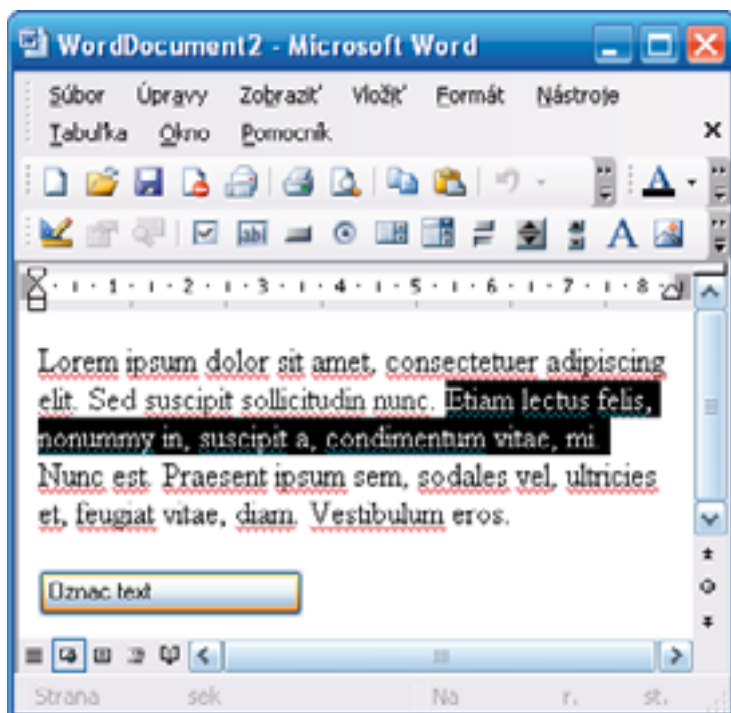
Nakoľko Word je textový editor, je niekedy užitočné výber bloku textu prostredníctvom prirodzených entít, na ktoré je možné text lexikálne rozčleniť. Napríklad môžeme vyberať bloky textu zadaním rozsahu viet. Začneme jednou vetou. Nasledujúci kód vyberie tretiu vetu.

### C#

```
private void btVyberVetu_Click(object sender, EventArgs e)
{
    Word.Range rng = this.Sentences[3];
    rng.Select();
}
```

### Visual Basic

```
Dim s2 As Word.Range = Me.Sentences(3)
s2.Select()
```



Výber tretej vety dokumentu

Vybranú vetu môžeme nahradiť iným textom

```
Word.Range rng = this.Sentences[3];  
rng.Select();  
rng.Text = „Novy text “;
```

alebo označiť dve vety a vymeniť ich texty

```
Word.Range rng1 = this.Sentences[3];  
Word.Range rng2 = this.Sentences[1];  
  
string sPom = rng1.Text;  
rng1.Text = rng2.Text;  
rng2.Text = sPom;
```

## Výber vety

Pri stanovení rozsahu viet označíme začiatok prvej vety bloku a koniec poslednej časti bloku. Kód pre označenie bloku od začiatku druhej po koniec tretej vety. Všimnite si, že v kóde je kontrolná podmienka aby text nebol kratší ako tri vety.

### C#

```
if (this.Sentences.Count >= 3)  
{  
    object startLocation = this.Sentences[2].Start;  
    object endLocation = this.Sentences[3].End;  
    rng = this.Range(ref startLocation, ref endLocation);  
    rng.Select();  
}
```

### Visual Basic

```
Dim rng As Word.Range  
  
If Me.Sentences.Count >= 3 Then  
    Dim startLocation As Object = Me.Sentences(2).Start  
    Dim endLocation As Object = Me.Sentences(3).End  
    rng = Me.Range(Start:=startLocation, End:=endLocation)  
    rng.Select()  
End If
```

## Pozície znakov na začiatku a konci intervalu

Dosiaľ sme rozsah bloku stanovili pomocou kódu. Primárne pri práci s dokumentom v textovom editore blok označí väčšinou používateľ. Pomocou kódu VSTO aplikácie dokážeme rozsah označeného bloku ľahko zistiť. V nasledujúcej ukážke kódu najskôr označíme blok v rozsahu jednej vety a následne zistíme rozsah označeného bloku výpisom počiatočnej a koncovej pozície.

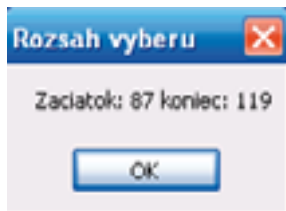
### C#

```
private void btVlozText_Click(object sender, EventArgs e)  
{  
    Word.Range rng = this.Sentences[3];  
    string startPosition = rng.Start.ToString();  
    string endPosition = rng.End.ToString();  
    MessageBox.Show(„Zaciatok: „ + startPosition + „ Koniec: „ + endPosition, „Rozsah vyberu“);  
}
```



## Visual Basic

```
Dim rng As Word.Range = Me.Sentences(2)
Dim startPosition As String = rng.Start.ToString()
Dim endPosition As String = rng.End.ToString()
MessageBox.Show(„Zaciatok: „ & startPosition & „ Koniec: „ & endPosition, „ Rozsah vyberu „)
```



Zobrazenie rozsahu výberu

## Výmena blokov

V jednej z predchádzajúcich statí je ukázaná vzájomná výmena obsahu dvoch dokumentov. Oveľa častejšia bude vzájomná výmena obsahu dvoch blokov na úrovni jedného dokumentu, samozrejme po malej modifikácii môžeme vytvoriť aj kód pre výmenu obsahu dvoch blokov z dvoch dokumentov.

## C#

```
Word.Range firstRange = this.Paragraphs[3].Range;
Word.Range secondRange = this.Paragraphs[5].Range;
```

```
string firstString = firstRange.Text;
string secondString = secondRange.Text;
```

```
firstRange.Text = secondString;
secondRange.Text = firstString;
```

```
firstRange.Select();
MessageBox.Show(firstRange.Text);
secondRange.Select();
MessageBox.Show(secondRange.Text);
```

## Visual Basic

```
Dim firstRange As Word.Range = Me.Paragraphs(3).Range
Dim secondRange As Word.Range = Me.Paragraphs(5).Range
```

```
Dim firstString As String = firstRange.Text
Dim secondString As String = secondRange.Text
```

```
firstRange.Text = secondString
secondRange.Text = firstString
```

```
firstRange.Select()
MessageBox.Show(firstRange.Text)
secondRange.Select()
MessageBox.Show(secondRange.Text)
```

## Prepísanie textu

Vzhľadom k primárnemu určeniu textového editora Word bude väčšina funkcií v tejto stati zameraná na prácu s textom. V tomto duchu sme začali aj úvodný príklad, cieľom ktorého bolo vloženie dokumentu. Po osvojení určenia miesta v texte a definovania intervalov dokážeme taktiež text na vybranom mieste wordového dokumentu prepísať.

```
private void btReplaceTextW_Click(object sender, EventArgs e)
{
    object start = 0;
    object end = 12;

    Word.Range rng = this.Range(ref start, ref end);
    rng.Text = „Prepisane“;
}
```

Okrem implicitného určenia rozsahu bloku môžeme vo vhodnom okamihu prepísať aj blok textu vybraný používateľom.

```
private void btReplaceTextW_Click(object sender, EventArgs e)
{
    object start = 0;
    object end = 12;

    Word.Range rng = this.Range(ref start, ref end);
    rng.Text = „Prepisane“;
}
```

## Pridanie komentára

Pokiaľ by sme mali čo i len stručne vymenovať pokročilejšie možnosti pre prácu VSTO aplikácie s wordovým dokumentom, rozsah tejto publikácie by prekročil niekoľko sto strán. Určitým vodítkom by nám mohli byť pokročilejšie funkcie wordu, napríklad sledovanie zmien, kontrola pravopisu, možnosť vkladania komentárov do dokumentu a podobne. Ako ilustračný príklad ukážeme kód pre vloženie komentára k určenému bloku wordového dokumentu.

### C#

```
object text = „Toto je komentar k prvemu odstavcu dokumentu.“;
this.Comments.Add(this.Paragraphs[3].Range, ref text);
```

### Visual Basic

```
Me.Comments.Add(Me.Paragraphs(1).Range, „ Toto je komentar k prvemu odstavcu dokumentu.“)
```



Vloženie komentára k odseku

Opačne, ak chceme z dokumentu vymazať všetky komentáre využijeme funkciu

## C#

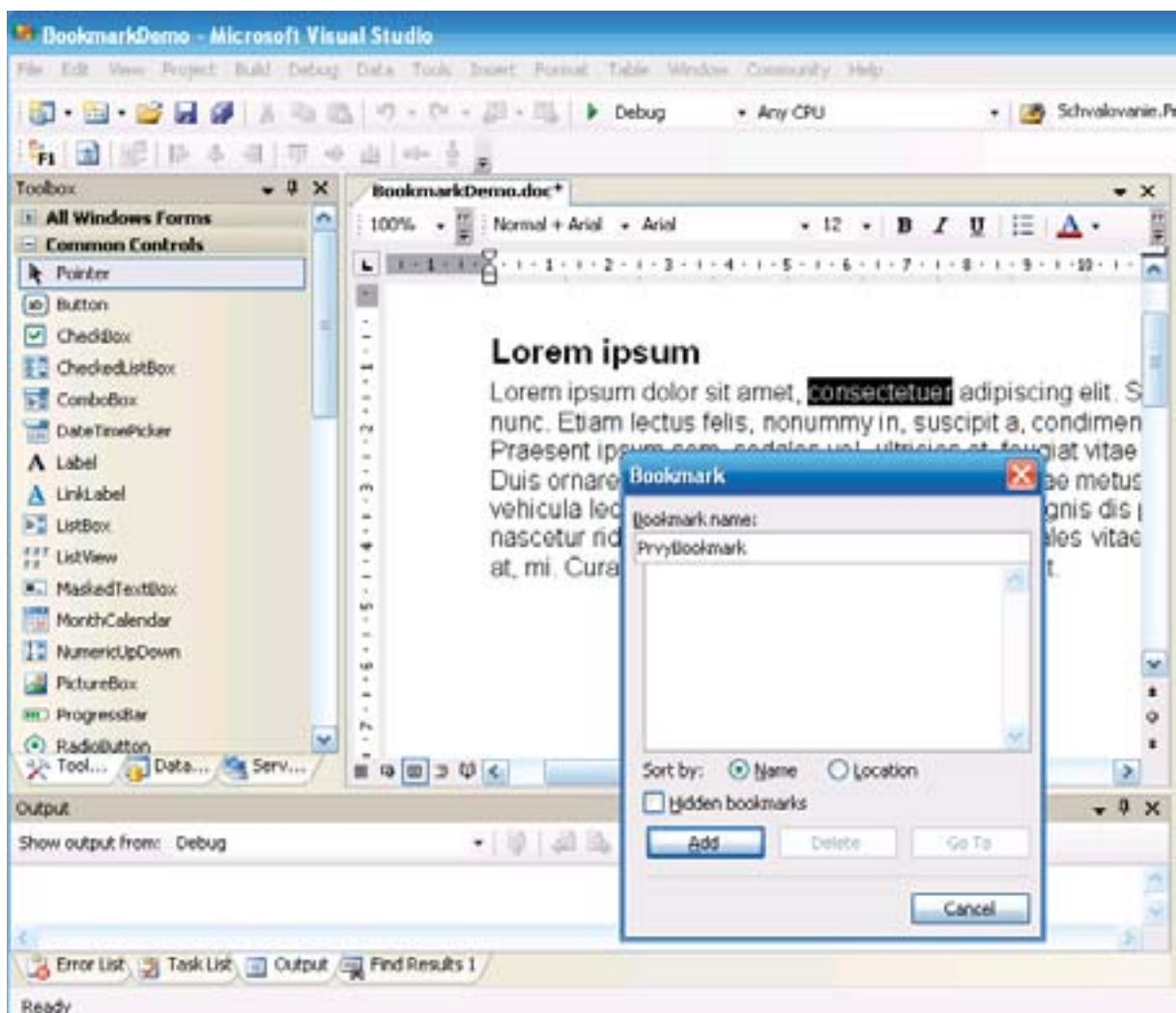
```
this.DeleteAllComments();
```

## Visual Basic

```
Me.DeleteAllComments()
```

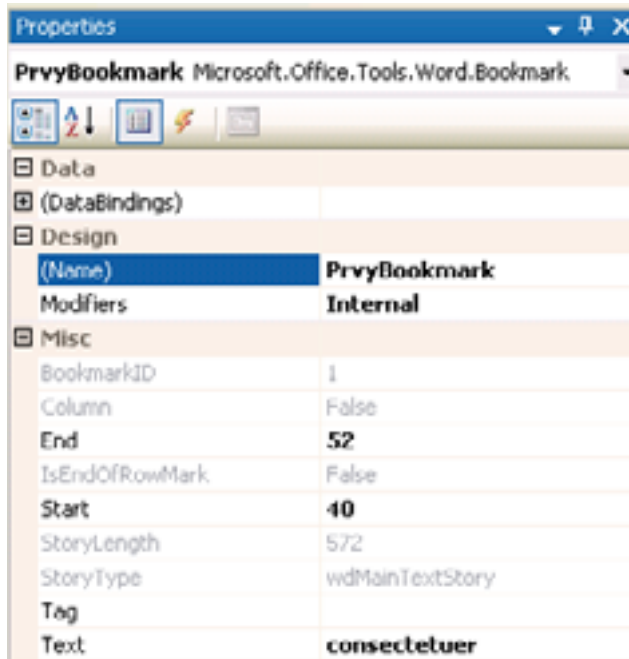
## Práca so záložkami (bookmarks)

Pri predchádzajúcich metódach označovania časti textu s ktorou chceme pracovať, ste určite postrehli určitú ťažkopádnosť. Dokážeme síce špecifikovať vety, alebo odseky, no najlogickejšie by bolo mať možnosť označiť časť textu priamo, pretože pre určenie príslušného rozsahu textu. Pre tento účel sa používajú vo Worde záložky (bookmarks). Záložky definujeme veľmi jednoducho, pravdepodobne ste s nimi vo Worde už pracovali, hlavne ak ste sa potrebovali operatívne orientovať v rozsiahlom texte.. Stačí označiť časť textu a aktivovať položku menu „Insert“, „Bookmark“. V dialógu pre špecifikovanie vlastnosti záložky zadáme jej názov.



Označenie časti textu Bookmarkom

Text označený viditeľnou záložkou bude zobrazený v sivých hranatých zátvorkách. Pomocou zaškrtávacieho políčka „Hidden bookmarks“ môžeme toto označenie skryť. Čo nám záložky umožňujú? Ak umiestnite vo okne Visual Studio kurzor na záložkou označený blok textu a kliknete naň, v okne Properties sa zobrazia jeho parametre. Môžeme skúsiť meniť parameter text. Všimnite si, že po potvrdení zmeny sa táto prejaví aj v dokumente.



Parametre bookmarku v okne Properties

Pre záložku môžeme programovým kódom reagovať na niektoré udalosti. Ak pridáme obslužnú procedúru na udalosť „Selected“, táto bude aktivovaná vtedy ak označený text alebo jeho časť bude vyselektovaná. Môžeme sa o tom presvedčiť napríklad výpisom oznamu.

```
private void PrvyBookmark_Selected(object sender, Microsoft.Office.Tools.Word.SelectionEventArgs e)
{
    MessageBox.Show(„Bookmark bol vybraty“);
}
```

Zaujímavou udalosťou, na ktorú potrebujeme občas reagovať je zmena výberu

```
private void DruhyBookmark_SelectionChange(object sender, Microsoft.Office.Tools.Word.SelectionEventArgs e)
{
    MessageBox.Show(„Bookmark bol zmeneny“);
}
```

Bookmarks majú najväčší význam pri vypĺňaní formulárových dokumentov, kedy ich šablónový obsah nahradíme príslušnými údajmi z databázových tabuliek.

## KAPITOLA 5: PANEL OVLÁDACÍCH PRVKOV

V predchádzajúcich kapitolách sme pri vysvetľovaní základných princípov a možností manipulácie s dokumentmi umiestňovali ovládacie prvky priamo do dokumentu. Niekedy, keď interaktívne pracujeme priamo s určitými pasážami dokumentu sa to hodí, inokedy je však takýto spôsob nevhodný, nakoľko ovládacie prvky zasahujú do dizajnu dokumentu. Preto VSTO aplikácie pre Office môžu obsahovať, presnejšie povedané dosť často obsahujú špeciálny ovládací panel (Actions Pane), zobrazený najčastejšie v pravej časti pracovnej obrazovky programov Word a Excel. Tento panel môžeme prirovnať v podstate k formuláru klasickej aplikácie a slúži ako kontajner pre ovládacie prvky.

V zásade môžeme v prípade výhodnosti alebo potreby migrovať na VSTO skoro akúkoľvek Win Form aplikáciu. Časť ovládacích prvkov môžeme umiestniť na panel ovládacích prvkov a niekedy aj celé formuláre je možné umiestniť do modálnych alebo nedomálnych dialógových okien, ktorých veľkosť môže v prípade potreby zaberáť aj celú obrazovku, takže sa do nich zmestia aj všetky ovládacie prvky z pôvodnej WinForm aplikácie, dokonca aj v pôvodnom rozložení.

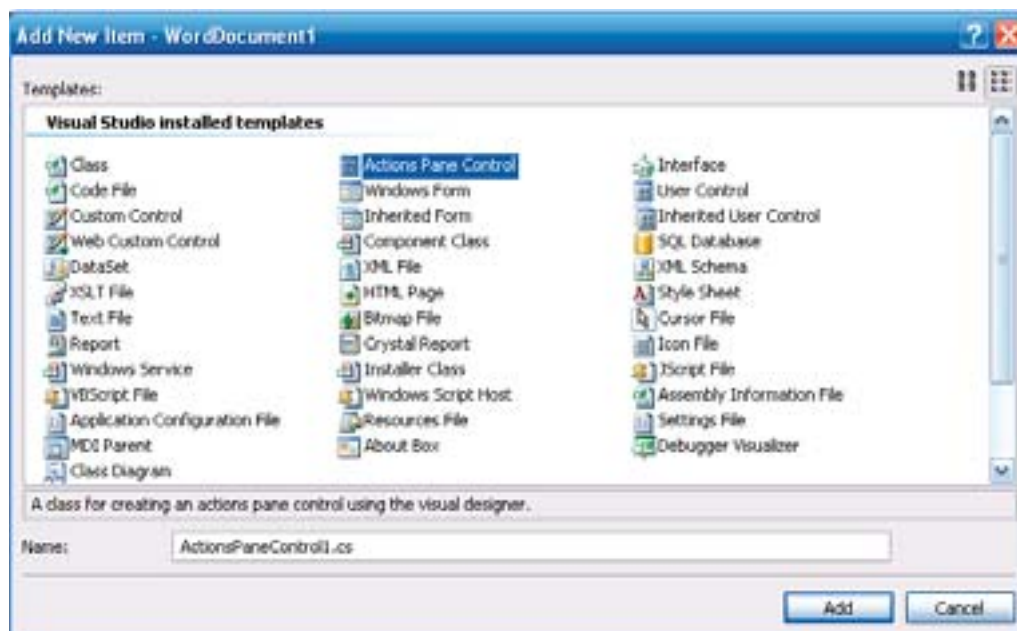
### „SmartDocuments“ ako predchodca „Actions Pane“

Ak trochu nahliadneme do histórie, predchodcom „Actions Pane“ bola technológia „Smart Documents“. V princípe sa jednalo o rozšírenie Office Task Pane o ďalšiu vlastnú funkcionálnu. Toto rozšírenie bolo nerozlučne späté s konkrétnym dokumentom. Pre jeho beh a vývoj bol potrebný doplnok SmartDocs SDK. Aplikácie bolo možné robiť vo vývojovom prostredí Visual Studio 6.0 alebo Visual Basicu 6.0.

Technológia „Actions Pane“ podobne ako jej predchodca „SmartDocuments“ rozširuje možnosti Office Task Pane, tentoraz však o možnosti.NET kódu.

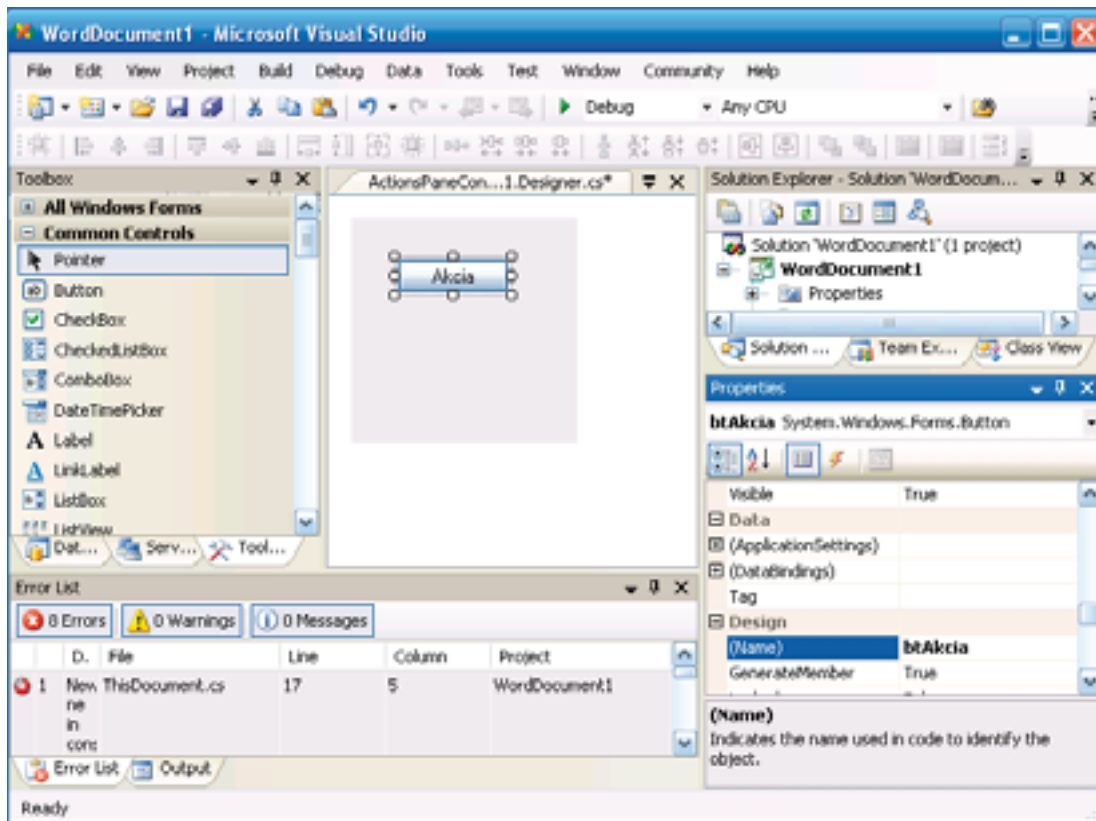
### Postup pridania panelu ovládacích prvkov do projektu

V menu „Project“, „Add New Item“ pridáme do projektu ovládací prvok Actions Pane Control.



Pridanie ovládacieho prvku Actions Pane Control

Vizuálne bude panel ovládacích prvkov v návrhovom zobrazení reprezentovaný sivým obdĺžnikom, ktorého rozmery je možné podľa potreby interaktívne meniť. Pri návrhu dizajnu a funkcionality ovládacieho panelu aplikácie sem umiestňujeme buď ovládacie prvky v návrhovom zobrazení, alebo sa tu zobrazia aj prvky vytvorené pomocou kódu dynamicky počas behu aplikácie. Ak by sme VSTO aplikáciu spustili v tomto okamihu, po pridaní panelu, nijaký panel by sa nezobrazil, a to ani v prípade ak by sme naň umiestnili nejaké prvky, napríklad tlačidlá, textboxy a podobne.



*Pridanie ovládacieho prvku Actions Pane Control*

Okrem súborov s údajmi o vizuálnych komponentách, vznikne v okamihu pridania panelu aj súbor pre obslužný kód. V okamihu návrhu je prázdny. Okrem „Using“ direktív obsahuje len kód pre inicializáciu prípadných vizuálnych prvkov.

### **ActionsPaneControl1.cs – Visual C#**

```
namespace WordDocument2
{
    partial class ActionsPaneControl1 : UserControl
    {
        public ActionsPaneControl1()
        {
            InitializeComponent();
        }
    }
}
```

### **ActionsPaneControl1.vb - Visual Basic**

```
Public Class ActionsPaneControl1

End Class
```

Pridaním panelu do projektu a jeho vizuálnym návrhom sa naša úloha ešte nekončí. Po spustení aplikácie by sme nijaký panel zatiaľ neuvideli. Panel je nutné pridať do projektu pomocou programového kódu. Najskôr v kóde `ThisDocument.cs` deklarujeme objekt typu `ActionsPaneControl`.

```
private ActionsPaneControl1 acp_panel;
```

a v obslužnej procedúre `ThisDocument_Startup` objekt vytvoríme a pridáme do aplikácie

```
acp_panel = new ActionsPaneControl1();  
this.ActionsPane.Controls.Add(acp_panel);
```

kompletný kód bude

### **ThisDocument.cs - Visual C#**

```
namespace WordDocument1  
{  
    public partial class ThisDocument  
    {  
        private ActionsPaneControl1 acp_panel;  
        private void ThisDocument_Startup(object sender, System.EventArgs e)  
        {  
            acp_panel = new ActionsPaneControl1();  
            this.ActionsPane.Controls.Add(acp_panel);  
        }  
  
        private void ThisDocument_Shutdown(object sender, System.EventArgs e)  
        {  
        }  
    }  
}
```

### **ThisDocument.cs - Visual Basic**

```
Public Class ThisDocument  
  
    Private apPanel As ActionsPaneControl1  
  
    Private Sub ThisDocument_Startup(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Startup  
        apPanel = New ActionsPaneControl1()  
        Me.ActionsPane.Controls.Add(apPanel)  
    End Sub  
  
    Private Sub ThisDocument_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Shutdown  
  
    End Sub  
  
End Class
```

Ak aplikáciu spustíme po týchto zmenách panel sa zobrazí buď prázdny, alebo v podobe ako sme ho navrhli v pravej obrazovke aplikácie.



## Ovládacie prvky na paneli

Na panel môžeme podľa potreby rozmiestniť potrebné ovládacie prvky z Toolboxu. Následne vytvorením kódu pre obsluhu udalostí s týmito prvkami súvisiacimi vybudujeme aplikačnú logiku. Napríklad po dvojkliku na symbol tlačidla sa vytvorí rutina pre obsluhu udalosti spojenej s prvkom. V prípade tlačidla je takouto udalosťou jeho zatlačenie.

```
private void btAkcia_Click(object sender, EventArgs e)
{
}
}
```

v súbore ActionsPaneControl1.Designer.cs je v tele inicializačnej metódy táto obslužná procedúra priradená k príslušnému ovládaciemu prvku

```
private void InitializeComponent()
{
    this.btAkcia = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // btAkcia
    //
    this.btAkcia.Location = new System.Drawing.Point(98, 152);
    this.btAkcia.Name = „btAkcia“;
    this.btAkcia.Size = new System.Drawing.Size(75, 23);
    this.btAkcia.TabIndex = 0;
    this.btAkcia.Text = „Akcia“;
    this.btAkcia.UseVisualStyleBackColor = true;
    this.btAkcia.Click += new System.EventHandler(this.btAkcia_Click);
    //
    // ActionsPaneControl1
    //
    this.Controls.Add(this.btAkcia);
    this.Name = „ActionsPaneControl1“;
    this.Size = new System.Drawing.Size(282, 471);
    this.ResumeLayout(false);
}
}
```

## Prístup k premenným v globálnom dokumente

Väčšina aplikácií potrebuje udržiavať určité informácie na globálnej úrovni, napríklad stavové premenné a podobne. Tieto údaje je výhodné mať na úrovni hierarchie globálneho dokumentu, čiže v súbore ThisDocument.cs. Napríklad ak na globálnej úrovni deklarujeme premennú pre počet prístupov a túto premennú naplníme.

```
namespace WordDocument1
{
    public partial class ThisDocument
    {
        public int nPocetPrístupov;

        private void ThisDocument_Startup(object sender, System.EventArgs e)
        {
            nPocetPrístupov = 9;
        }
        ...
    }
}
```

Z obslužného kódu panelu ovládacích prvkov získame k tejto premennej prístup

```
private void btAkcia_Click(object sender, EventArgs e)
{
    Globals.ThisDocument.nPocetPristupov += 1;
}
```

Rovnako dobre môžeme pomocou kódu aktivovaného z panelu ovládacích prvkov pracovať s obsahom hlavného dokumentu

```
private void btAkcia_Click(object sender, EventArgs e)
{
    Globals.ThisDocument.Paragraphs[1].Range.Text = „Hello World!";
}
```

## Zmena pozadia panelu

Nakoľko VSTO projekty sú určené pre prácu s dokumentmi, v mnohých prípadoch pomerne veľa záleží na celkovom dizajne aplikácie. K celkovému priaznivému dojmu môže prispieť aj vhodne ladené a štylizované pozadie ovládacieho panelu. Požadované obrázky skopírujeme do vhodného adresára projektu. V našom prípade sme vytvorili podadresár „Images“, do ktorého sme skopírovali tri súbory s obrázkami. Staticky nastavujeme obrázok pozadia pre panel pomocou parametra BackgroundImage. Obrázky je výhodné pridať medzi resource. Príslušný dialóg pre tento úkon nám bude pri nastavovaní obrázku pozadia ponúknutý automaticky.



Import obrázkov z podadresára do resourcov

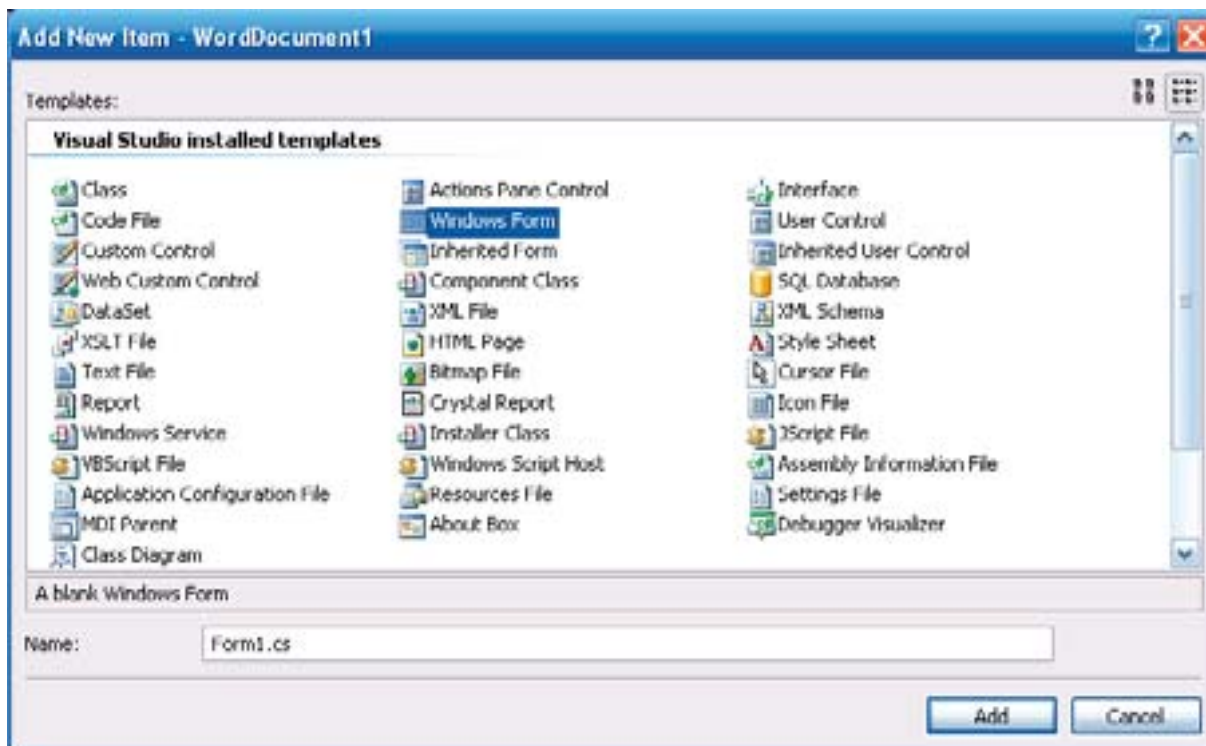
Pozadie panelu môžeme meniť aj počas behu aplikácie. V našom príklade sme na plochu aplikácie pridali dva tlačidlá, ktoré môžu prepínať funkčnosť aplikácie a pri tej príležitosti zmenia aj pozadie panelu. Obslužné procedúry zatlačenia príslušných tlačidiel obsahujú kód pre zmenu pozadia.

```
private void btPozadie1_Click(object sender, EventArgs e)
{
    this.BackgroundImage = SpravaDokumentov.Properties.Resources.referent;
}
```

```
private void btPozadie2_Click(object sender, EventArgs e)
{
    this.BackgroundImage = SpravaDokumentov.Properties.Resources.riaditel;
}
```

## Zobrazenie WinForm

Ako už bolo v úvode odstavca spomenuté veľmi silným nástrojom pri vytváraní používateľského rozhrania VSTO aplikácie, a s výhodou ich využijeme aj pri migrácii z WinForms aplikácie. Vizualne pridáme nový formulár do VSTO aplikácie z dialógového okna „Add New Item“



Vytvorenie nového Windows formulára

S formulárom pracujeme rovnako ako s dialógovým oknom.

```
FormPrihlas fPrihlas = new FormPrihlas();
```

```
fPrihlas.ShowDialog();
```

Globálne premenné môžeme mať v súbore ThisDocument.cs.

## KAPITOLA 6: PRÁCA S ÚDAJMI V DATABÁZACH

Aby vyvíjaná aplikácia naplnila ciele, pre ktoré bola vytvorená, musí poskytovať klientom dostatok informácií, a to v dnešnej dobe bez ohľadu na to, či sa jedná o personálnu aplikáciu a podobne. Požadované údaje musíme zhromaždiť a niekam uložiť. Jedným zo základných poslání VSTO aplikácie je generovať dokumenty a to na základe uložených a spracovaných údajov. Pre uloženie údajov máme niekoľko možností, jednou z nich je napríklad súborový systém. Ak však potrebujeme vyhľadávať informácie v súbore státisícov, prípade miliónov položiek, alebo poskytnúť utriedené údaje podľa rôznych kritérií, zistíme, že súborový systém ukladania údajov je veľmi neefektívny. Každú operáciu musíme sami prácne naprogramovať a rýchlosť prístupu nás málokedy uspokojí. A čo je dôležité, väčšie množstvo údajov ešte samo o sebe neznamená vyššiu kvalitu. Dôležité je, ako bezpečne a spoľahlivo sú údaje uložené a aký rýchly je k nim prístup. Oveľa efektívnejšou technológiou pre ukladanie a správu údajov sa javí databáza pod správou databázového servera. Databázu chápeme ako úložisko údajov, ktoré sú uložené a spracovávané nezávisle od aplikačných programov. Databázy zapuzdrujú jednak vlastné údaje, ale aj relačné vzťahy medzi jednotlivými prvkami a objektmi v databáze, schémy popisujúce štruktúry údajov a integritné obmedzenia.

Pri použití databázy nám stačí definovať požadovanú operáciu nad množinou údajov v databáze pomocou príkazu jazyka SQL (Structured Query Language) a ostatné už vykoná databázový program sám. Takýto program nazývame databázový server. Jeho práca je spravidla veľmi efektívna, pretože údaje sú priebežne indexované a databáza sa sama optimalizuje, prípadne sa opravujú poškodené údaje. VSTO aplikácie cez vhodné rozhranie dokážu efektívne pracovať prakticky s každým databázovým serverom.

Databázové aplikácie vyvíjané pre Microsoft Word a Excel využívajú takzvaný Data – View model, kedy pohľady na údaje a údaje sú od seba hierarchicky oddelené tak, že údaje sú napojené na pohľady, prostredníctvom ktorých sú vo vhodnej forme prezentované navonok. Pohľady (Views) sú akési kontajnery pre údaje. Ak si pripomenieme architektúru Wordovej aplikácie z tretej kapitoly, takýmito pohľadmi môžu byť napríklad globálne objekty Document (Worksheet u Excel aplikácie) prípadne vizuálne prvky, napríklad TextBox.

Pohľad dokáže údaje len zobrazit', neumožňuje nad nimi vykonávať žiadne iné operácie. K jednému dátovému zdroju môžeme mať pripojených viac rôznych pohľadov.

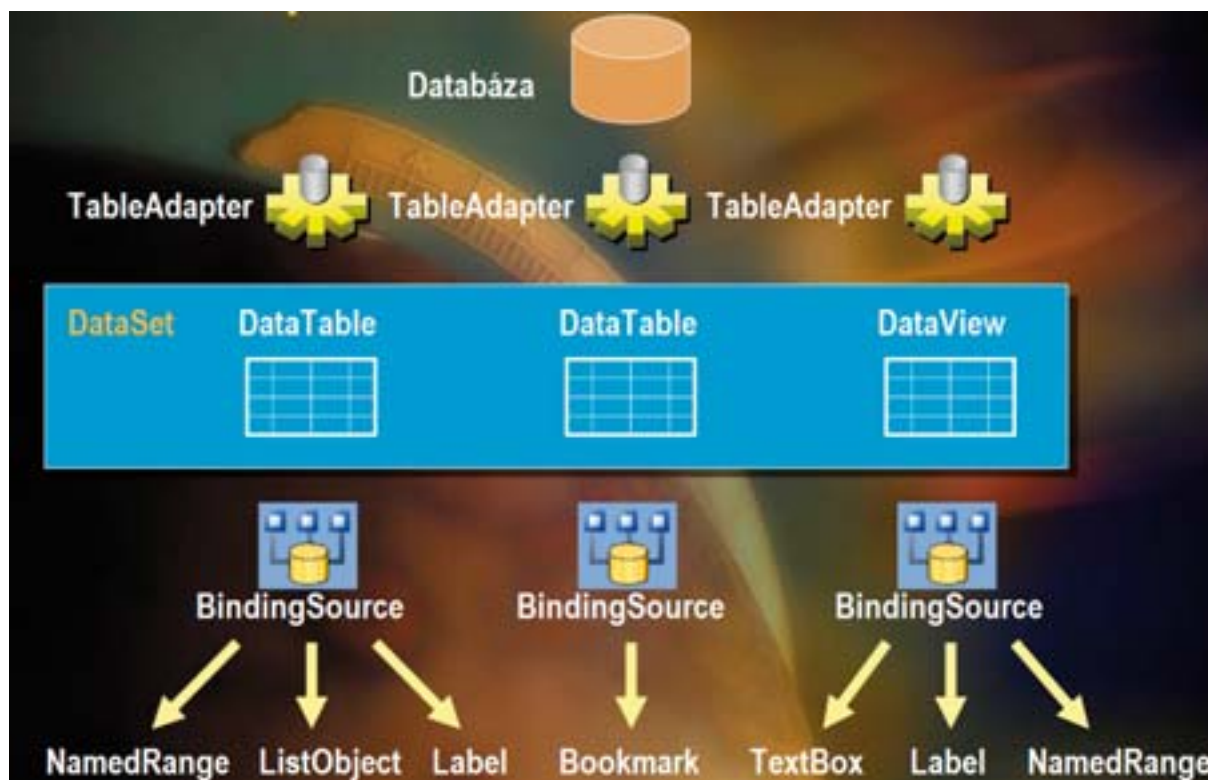


Schéma Data – View modelu pre prístup VSTO aplikácie k údajom

Východisková situácia pre migráciu na vývoj databázových VSTO aplikácií je výhodnejšia pre vývojárov WinForms, alebo ASP.NET 2.0 aplikácií. Pre prácu s údajmi vo VSTO aplikáciách nepotrebujeme poznať objektový model Office 2003, musíme však poznať technológiu ADO.NET 2.0.

## Prístup k údajom SQL Servera 2005 cez klientskú konzolovú aplikáciu

Ak chceme robiť pokusy s údajmi v databázach, vyvstane hlavne pre začiatočníkov niekoľko otázok a problémov. Napríklad, či požadované údaje v databáze už sú, prípadne ak nie sú ako ich tam dostať, ako ich vypísať, zmeniť, skontrolovať, či naša aplikácia pracuje s údajmi tak ako požadujeme. Hlavne na začiatku to môže byť začarovaný kruh. Niektoré aplikácie už pri svojom štarte predpokladajú, že v databáze budú nejaké údaje. Inokedy potrebujeme aplikáciu ladiť s testovacou vzorkou údajov.

Preto skôr než sa prepracujeme k vývoju VSTO databázovej aplikácie, predstavíme pre začiatočníkov aspoň v hrubých rysoch možnosti práce s databázovým serverom SQL Server 2005.

Ak by sme to mali zovšeobecniť, pre správu databázového serveru a ladenie databázovej časti aplikácií potrebujeme dva typy klientských aplikácií. SQL príkazy môžeme pred ich zakomponovaním do VSTO kódu ladiť pomocou **klientskej konzolovej aplikácie**. Náplň jej činnosti je jednoduchá. Služi pre zadávanie príkazov jazyka SQL databázovému serveru a okne tej istej aplikácie taktiež vidíme výstupy, ktoré databázový server vygeneruje ako odozvu na naše príkazy, teda napríklad výpis obsahu databázových tabuliek, potvrdenie vykonania našich príkazov, chybové hlásenia a podobne. Pre administráciu databázy, napríklad pre nastavovanie prístupových práv potrebujeme **aplikáciu pre správu databázy**. Pomocou nástroja pre správu databázy je možné nastaviť stratégiu údržby a zálohovania údajov v databáze a podobne. Príkazy pre vytvorenie a naplnenie databázových štruktúr z predchádzajúcej state sú univerzálne (možno s nepatrnými úpravami) pre všetky databázové platformy. Znovu zdôrazňujeme, že je technicky jedno akú databázovú platformu použijeme. Pre zadávanie údajov však musíme použiť vhodnú klientskú konzolovú aplikáciu.

## Prístup k databáze cez SQL Server Management Studio

SQL Server Management Studio je komplexné integrované prostredie pre správu databázového serveru SQL Server 2005. Ovládanie tejto konzoly je pre vývojárov veľmi intuitívne, nakoľko Management Studio je vybudované na základe unifikovaného vývojového prostredia Microsoft Development Environment, ktoré vychádza z vývojového prostredia Visual Studio 2005.

Nástroj sa spúšťa štandardným spôsobom z menu operačného systému Windows *Start | All Programs | Microsoft SQL Server 2005 | SQL Server Management Studio*. Po spustení nástroja sa zobrazí dialóg pre pripojenie sa k databázovému serveru. Ak robíme prvé pokusy na lokálnom alebo virtuálnom počítači, môžeme nastaviť názov serveru ako localhost a ako metódu prihlásenia použiť Windows Authentication.



Prihlasovací dialóg SQL ServerManagement Studia

Pracovná plocha SQL Management Studia je rozdelená na niekoľko častí.

## Registered Servers

Okno obsahuje zoznam serverov, na ktoré sa možno pomocou Management Studia pripojiť.

## Object Explorer

Toto okno je veľmi podobné ľavému panelu Enterprise Managera u staršej verzie SQL Servera 2000 a poskytuje grafický, prehľadný, hierarchický pohľad na SQL komponenty až na úroveň stĺpcov a indexov. Je potrebné si uvedomiť, že objekty sú vo verzii SQL Serveri 2005 asociované so schémou a nie s vlastníkom ako u predchádzajúcej verzie SQL 2000.

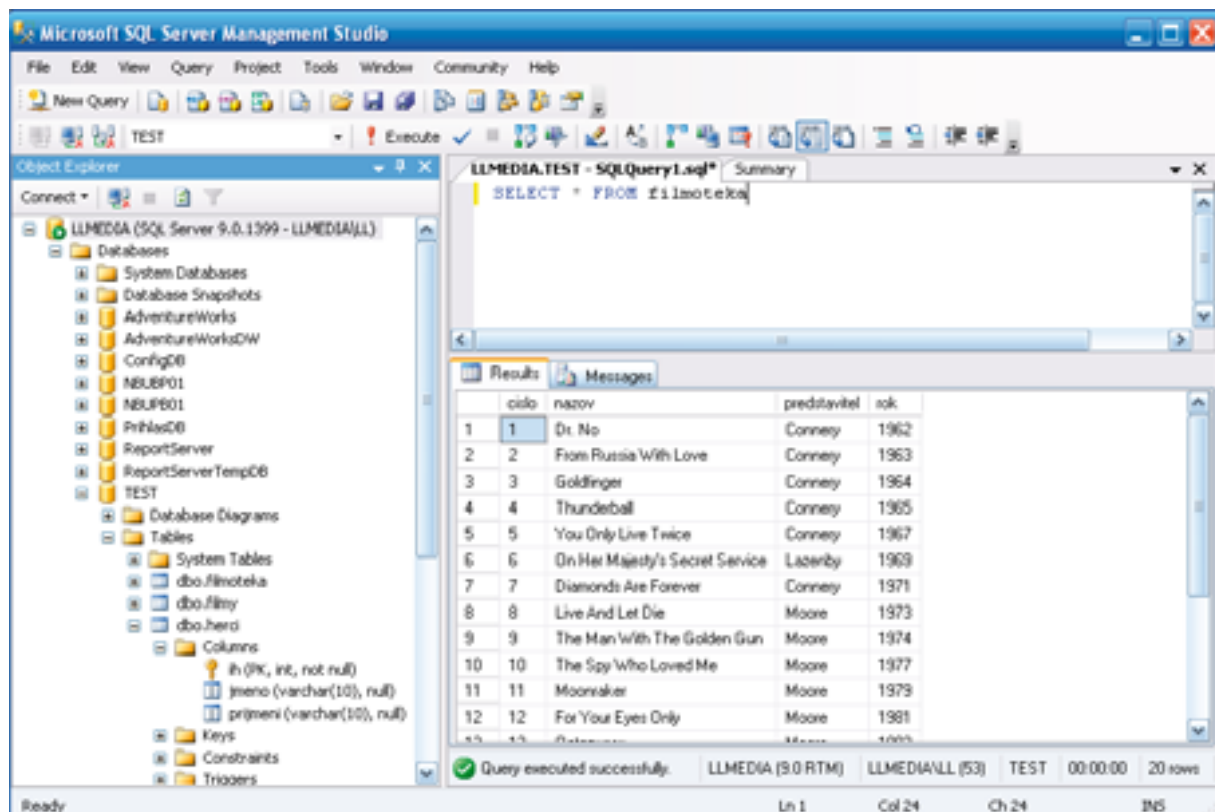
## Hlavné okno v strede pracovnej plochy

Po spustení aplikácie je možné hlavné okno pracovnej plochy prepnúť do režimov

- List
- Report

V záložke Report môžeme zvoliť, či chceme sledovať štatistiky prístupov na disk, transakcií, použitie indexov a podobne

Po aktivovaní tlačidla New Query aktivujeme dopytovací mód Management Studia. Tento mód je nástupcom samostatného nástroja Query Analyzer zo staršej verzie SQL Server 2000.



SQL Server Management Studio v režime dopytovania



Skôr než začneme zadávať SQL dopyty do niektorej databázy, potrebujeme preskúmať, či je Management Studio prepnuté na správnu databázu. Prepnutie sa realizuje buď pomocou combo boxu na hornom toolbare aplikácie, alebo príkazom.

*use nazov\_databazy*

napríklad

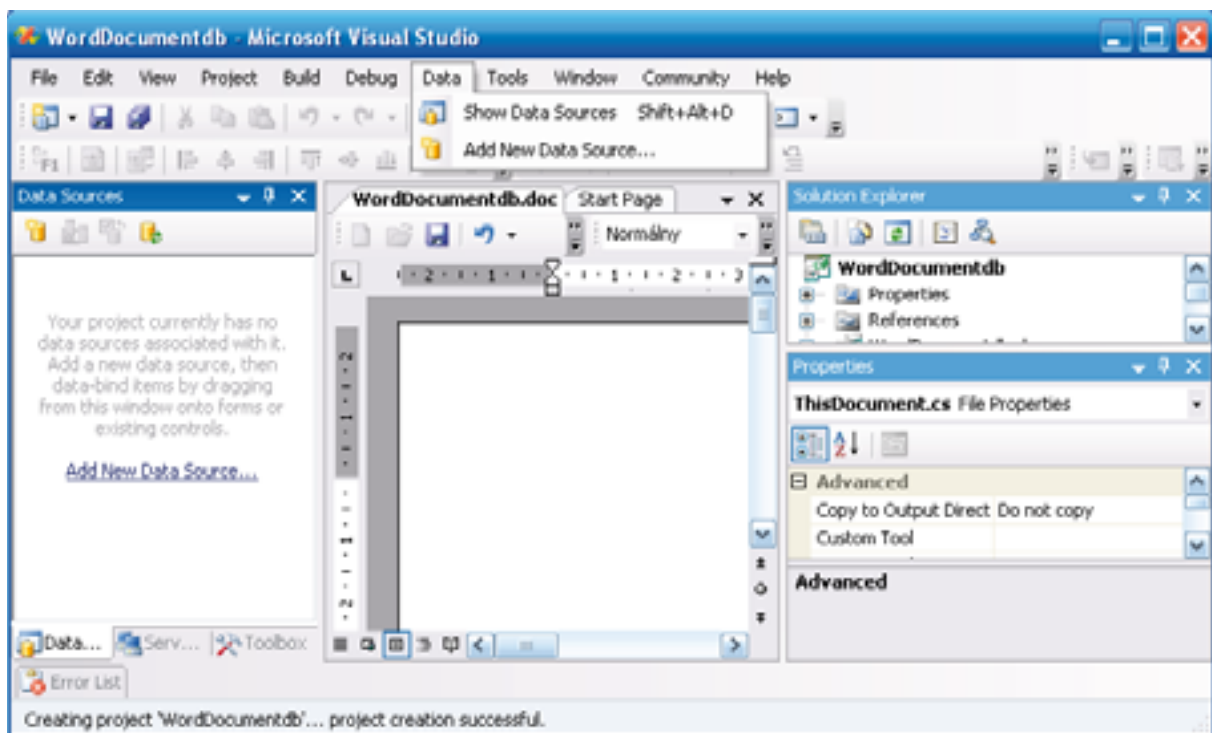
*use TEST*

Ak máme správne prepnuté na databázu TEST, môžeme položiť jednoduchý dopyt, napríklad

*SELECT \* FROM pokus;*

## Práca s údajmi priamo vo Visual Studiu

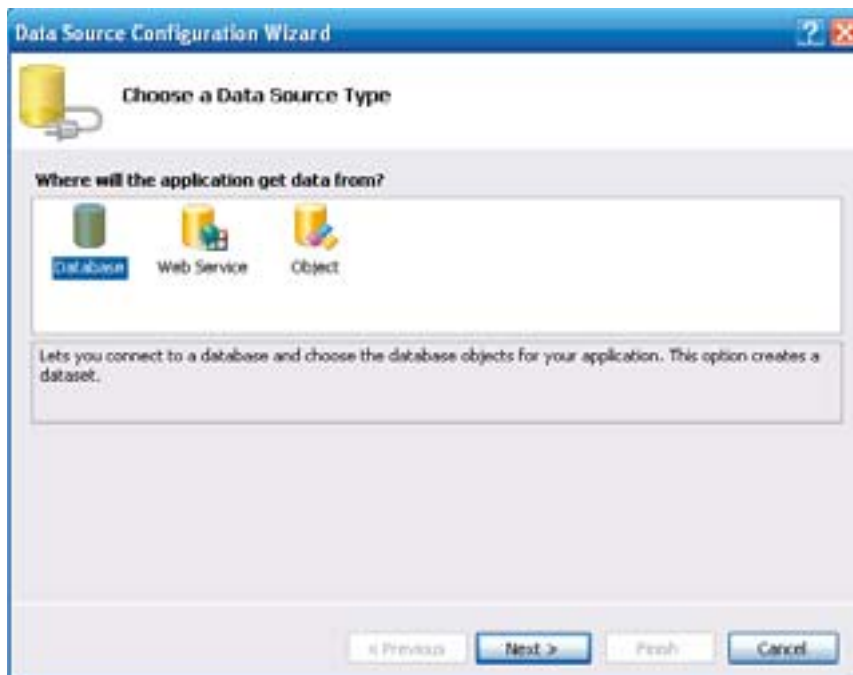
Najjednoduchšie sa k databáze pripojíme pomocou sprievodcu pripojenia sa k dátovým zdrojom. Pre zobrazenie dátových zdrojov aktivujeme položku menu Data – Show Data Sources. Okno s dátovými zdrojmi sa zobrazí v ľavom stĺpcovom okne vývojového prostredia. Pomocou menu Data – Add New Data Source, prípadne pomocou rovnomeného linku v okne Data Sources môžeme vytvoriť pripojenie k dátovému zdroju.



*Menu pre zobrazenie dátových zdrojov*

V prvom kroku volíme typ dátového zdroja. Môže to byť databáza, webová služba alebo objekt generujúci údaje.





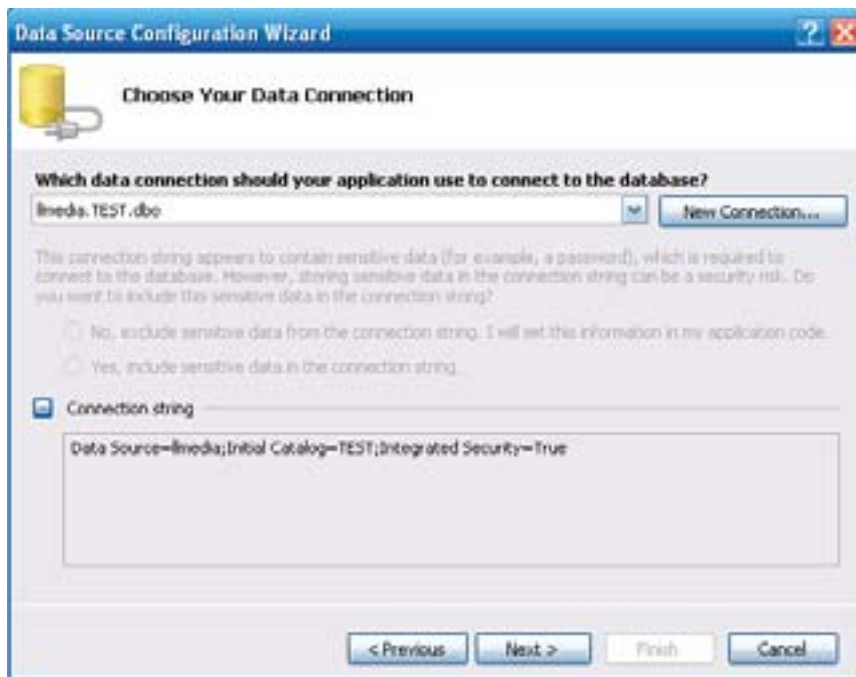
Výber typu dátového zdroja

V prípade ak vyberieme ako dátový zdroj databázu, je potrebné definovať parametre pre pripojenie sa k príslušnému databázovému serveru. Technicky povedané – pomocou návrhového dialógu vytvoríme pripojovací reťazec. Ak nestanovíme inak bude pripojovací reťazec uložený do konfiguračného súboru app.config, ktorý sa vytvorí v adresári projektu aplikácie.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
  </configSections>
  <connectionStrings>
    <add name="WordDocumentdb.Properties.Settings.TESTConnectionString"
      connectionString="Data Source=llmedia;Initial Catalog=TEST;Integrated Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

Konfiguračný reťazec pre pripojenie sa k zdroju údajov vznikne na základe vizuálneho návrhu. Môžeme ho zmeniť, napríklad pri vývoji a testovaní aplikácie môžeme využívať lokálnu databázu na vývojárskom počítači a pri nasadení na reálny server sa po zmene pripojovacieho reťazca môžeme pripojiť k reálnemu databázovému serveru.

```
<add name="Potvrdenie.Properties.Settings.REPConnectionString"
  connectionString="Data Source=llmedia;Initial Catalog=REP;Persist Security Info=True;User
  ID=user1;Password=Pass@word1"
  providerName="System.Data.SqlClient" />
</connectionStrings>
```



Menu pre zobrazenie dátových zdrojov

Po definovaní parametrov pripojenia pokračujeme vo vývoji databázovej časti aplikácie výberom databázových objektov. Môžeme z predmetnej databázy vyberať databázové tabuľky, pohľady, uložené procedúry alebo funkcie vracajúce údaje. U rozsiahlejších tabuliek a pohľadov môžeme prejsť na jemnejšiu úroveň granularity a vyberať jednotlivé stĺpce, tie ktoré potrebujeme pre generovanie dokumentov.

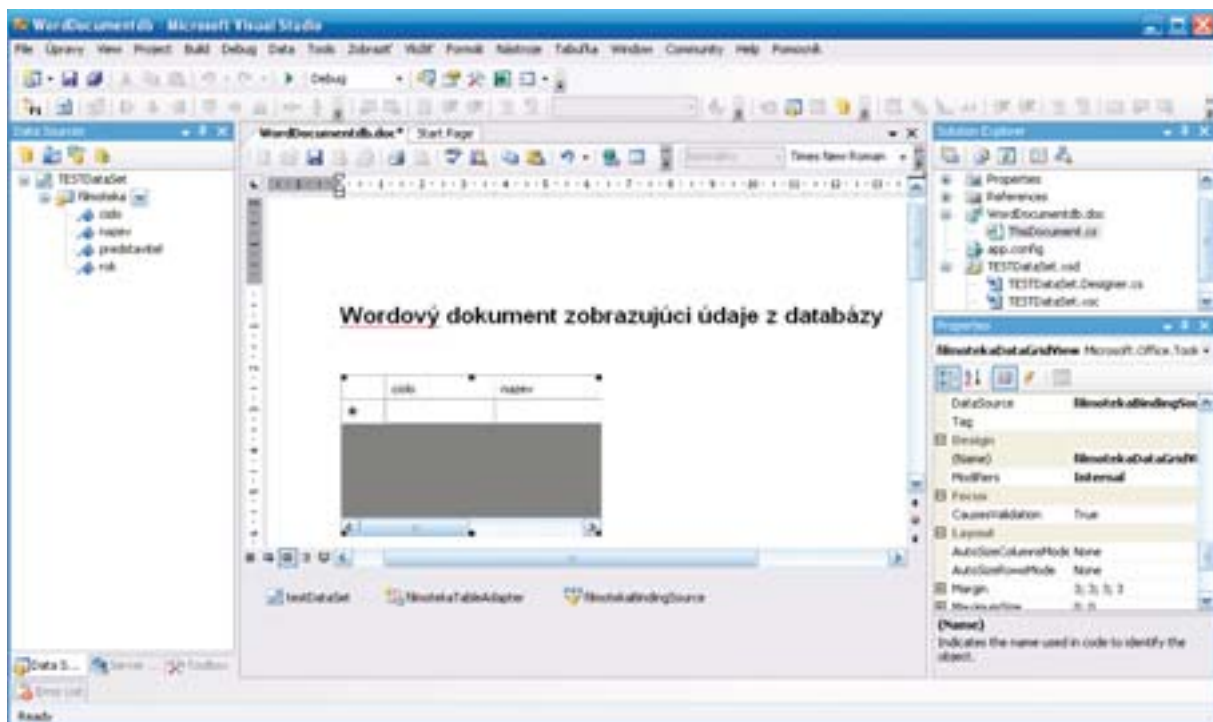


Výber databázových objektov

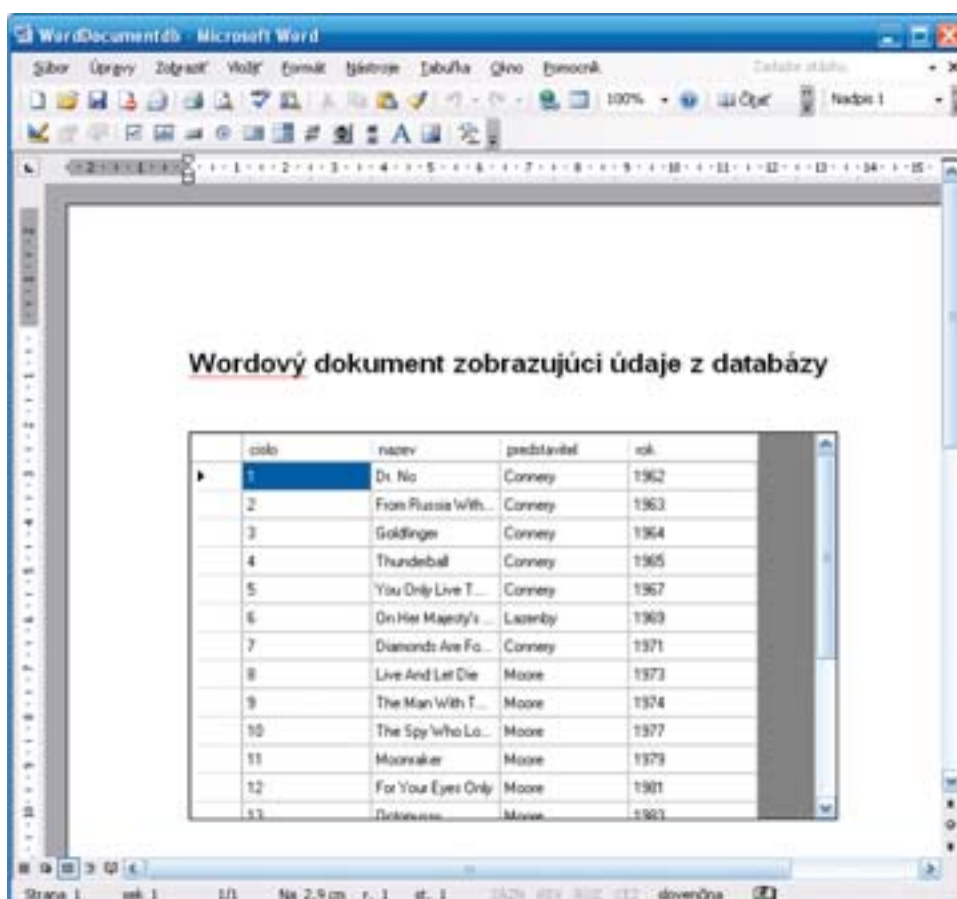
Azda najjednoduchším úkonom je presun celej databázovej tabuľky na plochu dokumentu, či už sa jedná o dokument programu Word, alebo hárok programu Excel. Počas tohto triviálneho úkonu sa vytvoria tri typy objektov.

- typovo silný dataset
- TableAdapter
- BindingSource

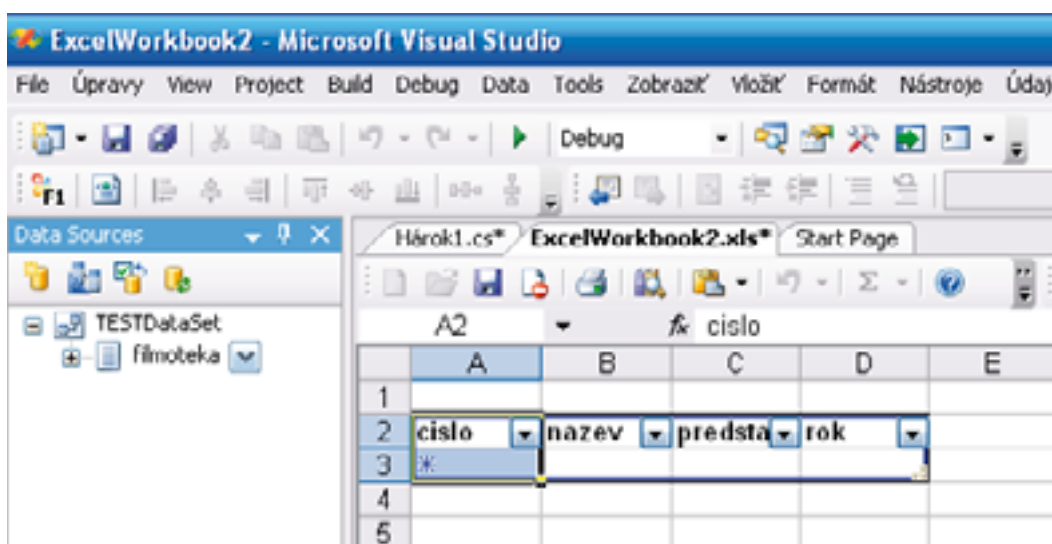
Z hľadiska návrhu bude do projektu pridaný vizuálny prvok Data Grid View. Tento sa zobrazí na mieste, na ktoré sme presunuli vizuálny symbol tabuľky. Presun môžeme vykonať buď priamo na plochu dokumentu (Word, Excel...), alebo na panel ovládacích prvkov, prípadne na samozatný Win Forms formulár. Ak umiestňujeme vizuálne zobrazenie databázovej tabuľky na panel ovládacích prvkov, môžeme podľa charakteru zobrazovanej tabuľky zvoliť zvislú alebo vodorovnú orientáciu panelu ovládacích prvkov. Pre širšiu tabuľku teda môžeme umiestniť panel vodorovne, do hornej, alebo dolnej časti formulára. Pre užšiu tabuľku v ktorej chceme zobrazit' viac riadkov zvolíme zvislé usporiadanie panelu v pravej alebo ľavej časti formulára.



*Presun databázovej tabuľky do dokumentu*



Vzhľad wordového dokumentu s databázovou tabuľkou



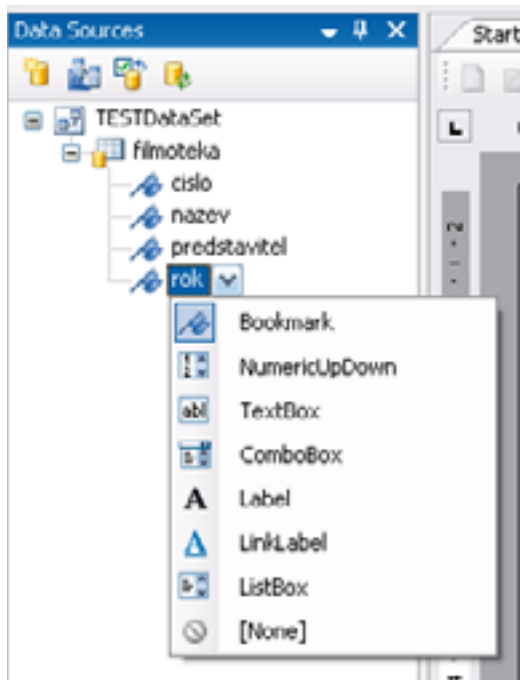
Presun databázovej tabuľky do hárku programu Excel

Vizuálnym presunom symbolu tabuľky alebo pohľadu získame jednoduché a efektné riešenie pre zobrazenie údajov z databázovej tabuľky, prípadne viacerých relačne zviazaných tabuliek, no pravdepodobne len dovedty, kým to nebudeme potrebovať použiť prakticky. Vtedy sa musíme prispôsobiť flexibilným požiadavkám a vytvoriť formulár alebo report podľa toho ako je potrebné z hľadiska aplikačnej logiky, legislatívy, politiky vedenia písomnosti. Vizuálny návrh presunom symbolov umožňujú preniesť objekt nielen ako tabuľku, ale aj ako detail. Stačí kliknúť pravým tlačidlom myši na symbol gridu vedľa názvu filmoteka. K dispozícii máme možnosť prepnutia do režimu „Detail“.

Ak v okne Data Sources rozvinieme tabuľku na úroveň jednotlivých stĺpcov a klikneme pravým tlačidlom na ikonu niektorého stĺpca, zobrazí sa kontextové menu, pomocou ktorého môžeme stanoviť vo forme akého ovládacieho prvku umiestnime príslušný stĺpec na plochu dokumentu. Môže byť umiestnený ako

- Bookmark
- Numeric UpDown
- TextBox
- ComboBox
- Label
- Link Label
- Listbox

Z takto nastavených stĺpcových ovládacích prvkov potom zostavíme formulár alebo report



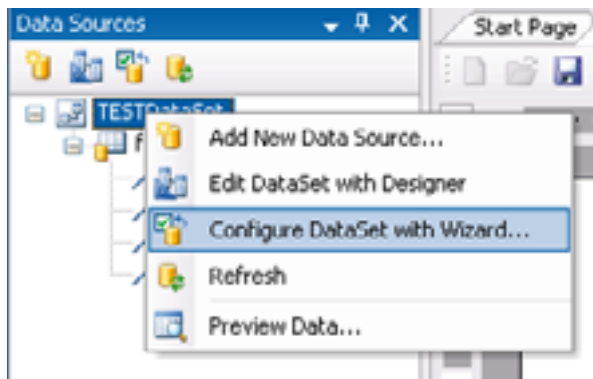
*Možnosti pre pridávanie jednotlivých stĺpcov vo forme rôznych ovládacích prvkov*

**Wordový dokument zobrazujúci údaje z databázy**

nazev:	<input type="text" value="Dr. No"/>
predstavitel:	<input type="text" value="Connery"/>
rok:	<input type="text" value="2"/>

*Formulár vytvorený z ovládacích prvkov*

Datový zdroj môžeme nakonfigurovať alebo prekonfigurovať aj neskôr a to na dvoch úrovniach. Pomocou položky menu „Configure DataSet With Wizard“ sa znovu dostaneme do dialógu pre výber objektov, teda tabuliek, pohľadov, uložených procedúr alebo funkcií, prípadne stĺpcov tabuliek.

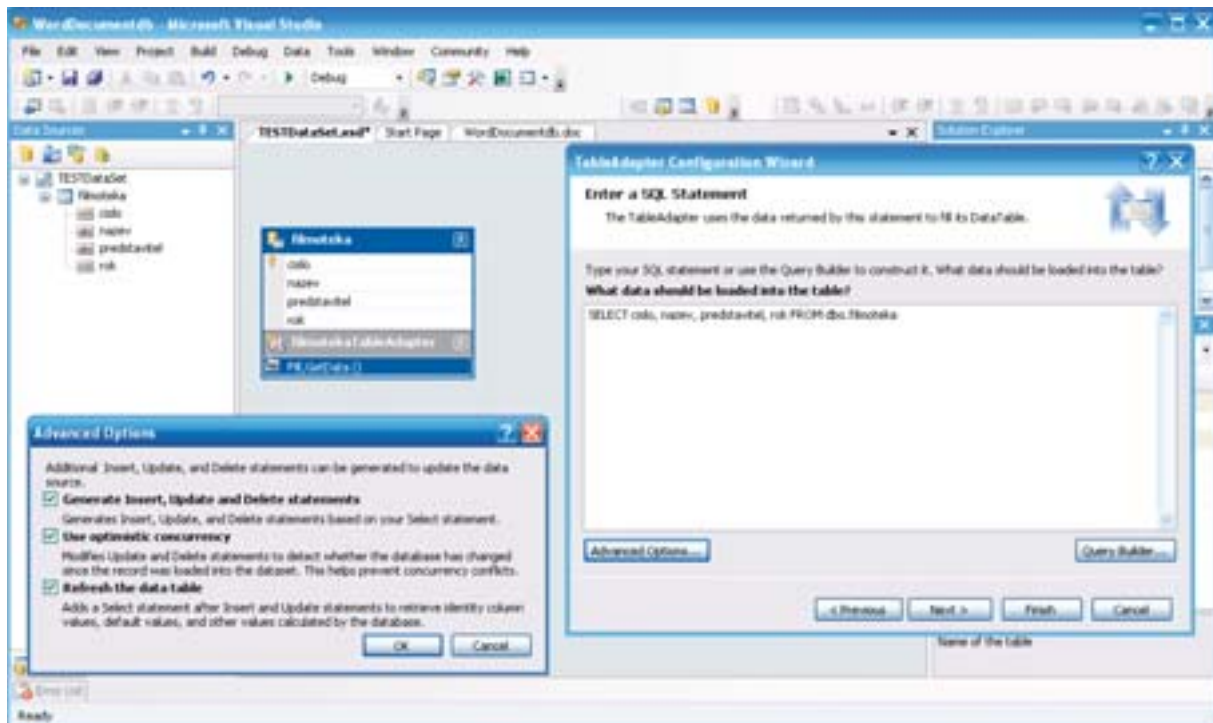


*Možnosti pre úpravu zdroja údajov*

Konkrétnu konfiguráciu data setu môžeme spresniť pomocou položky menu „Edit DataSet with Designer“.

## TableAdapter

Predstavuje komunikačnú vrstvu medzi datasetom a databázou. Pre každú tabuľku je vygenerovaný samostatný TableAdapter. Objekt TableAdapter je možné použiť pre naplnenie objektu DataTable v datasete, ukladanie zmien do dátového zdroja a spúšťanie uložených procedúr. Konfiguráciu TableAdapteru nám značne zjednoduší sprievodca TableAdapter Configuration Wizard.



*Možnosti pre editovanie DataSetu*



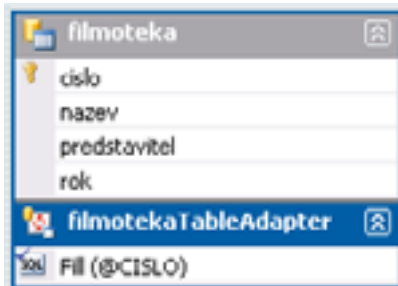
Pre výber údajov je použitý jednoduchý alebo parametrický dopyt. V okne TableAdapter Configuration Wizard máme pôvodný dopyt pre výber požadovaných stĺpcov, napríklad

```
SELECT cislo, nazev, predstavitel, rok FROM dbo.filmoteka
```

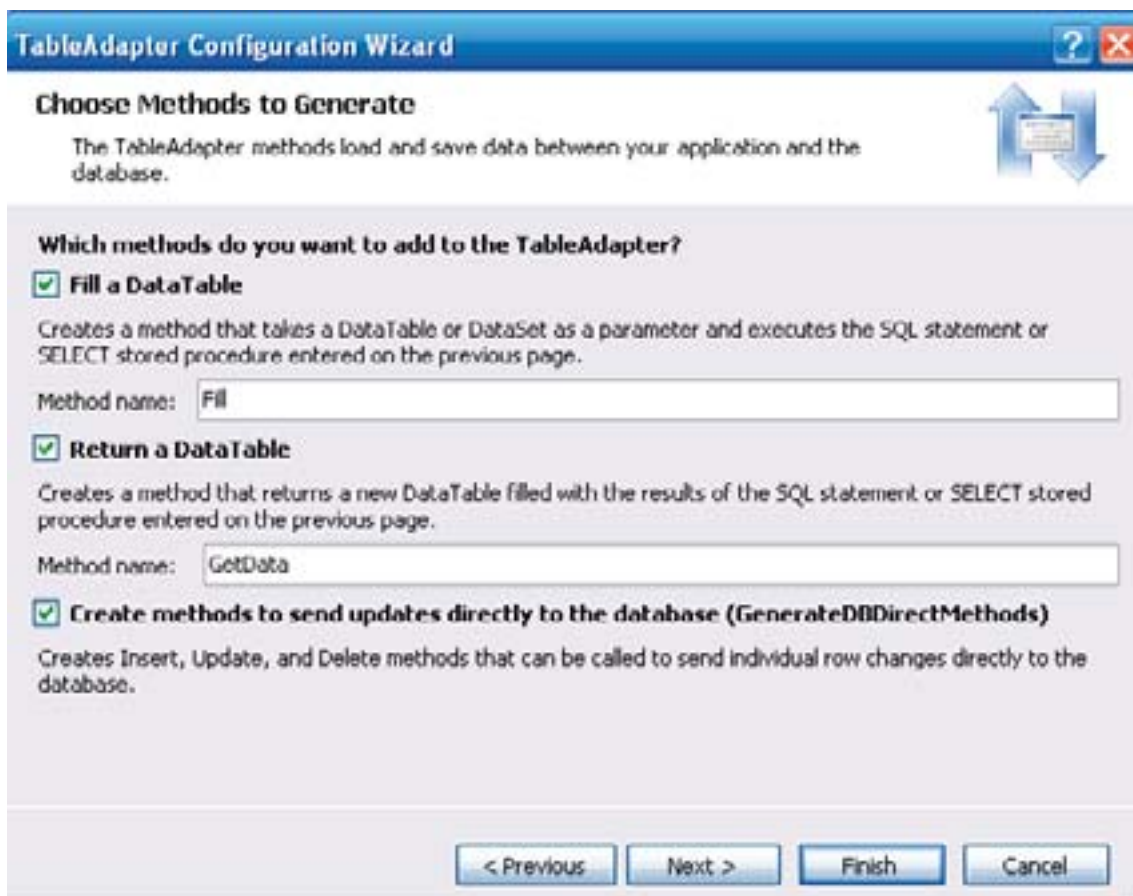
Po pridaní parametrov bude dopyt v tvare

```
SELECT cislo, nazev, predstavitel, rok FROM dbo.filmoteka WHERE cislo = @CISLO
```

Pridaním parametra sa zmení aj volanie metódy Fill(). Metóda sa bude volať s pridanými parametrami



Metóda Fill s parametrom



TableAdapter Configuration Wizard



Ak zavedieme parametrický dopyt, tým pádom musíme volať aj metódu Fill() s parametrom alebo parametrami. Implicitne je metóda Fill pre naplnenie datasetu volaná v tele metódy

```
private void ThisDocument_Startup(object sender, System.EventArgs e)
{
    // TODO: Delete this line of code to remove the default AutoFill for 'testDataSet.filmoteka'.
    if (this.NeedsFill(„testDataSet“))
    {
        this.filmotekaTableAdapter.Fill(this.testDataSet.filmoteka);
    }
}
```

Po doplnení parametru bude volanie v tvare

```
this.filmotekaTableAdapter.Fill(this.testDataSet.filmoteka, 3);
```

kde 3 je konkrétna hodnota parametra

## BindingSource

Pripojenie sa k zdroju údajov môžeme realizovať buď ako jednoduché, alebo komplexné. V prípade jednoduchého pripojenia spájame hodnotu z databázovej tabuľky s vlastnosťou, spravidla textovým obsahom nejakého vizuálneho ovládacieho prvku, napríklad text boxu. Pomocou komplexného pripojenia realizujeme načítanie viacerých hodnôt buď z databázovej tabuľky, alebo trebárs z XML súboru.

Objekt BindingSource poskytuje metódy pre základnú navigáciu v množine údajov: MoveFirst(), MoveNext(), MovePrevious(), MoveLast() a metódy pre manipuláciu s údajmi: Find(), Filter(), Sort(), AddNew(), a Remove(). Počas návrhu aplikácie nastavujeme vlastnosť DataBindings, počas behu aplikácie sa volá metóda DataBindings.Add()

Je potrebné si uvedomiť, že všetky vykonané zmeny zostávajú len na úrovni datasetu. Aby sa premietli aj do databázy, je potrebné použiť objekt TableAdapter, konkrétne jeho metódu Update().

Vzhľadom na rozdielnosť koncepcie programov Word a Excel sú rozdielne metódy, ako údaje z databázy zobrazíť prirodzenými zobrazovacími prostriedkami daného programu. U Wordu je to text, u Excelu tabuľka. Väčšinou potrebujeme údajmi nahradiť nie celý dokument, ale v prípade Wordu označené časti textu alebo šablóny (záložka, bookmark), v prípade Excelu v pomenovanom rozsahu buniek

## Bookmark (Visual Basic)

```
FirstNameBookmark.DataBindings.Add ("Value2", Me.EmployeesBindingSource, "FirstName", formattingEnabled, updateMode)
```

## NamedRange (Visual Basic)

```
FirstNameNamedRange.DataBindings.Add ("Value2", Me.EmployeesBindingSource, "FirstName", formattingEnabled, updateMode)
```

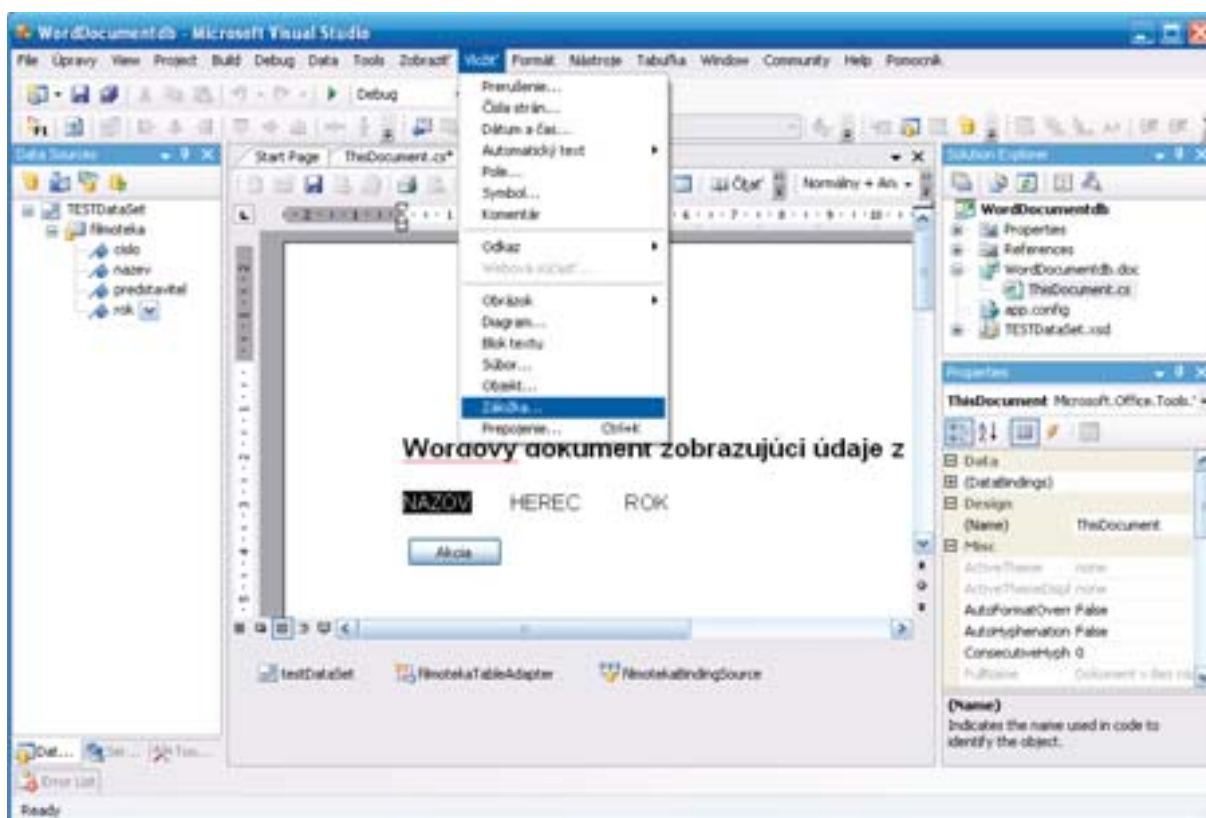
Ukážeme to na konkrétnych príkladoch, pre zmenu v C#

## Zviazanie záložky s údajom z databázy v programe Word

Objekt Bookmark tvorí spojitú oblasť dokumentu ohraničenú pozíciou prvého a posledného znaku. Pozície sú definované pri vytváraní záložky. Označený text môžeme pomocou menu Vložiť – Záložka určiť ako objekt Bookmark.

Zviazanie konkrétneho objektu Bookmark môžeme urobiť buď v návrhovom zobrazení, prenesením symbolu databázového stĺpca na symbol bookmarku alebo priamo pomocou kódu

```
private void btAkcia_Click(object sender, EventArgs e)
{
    bmHEREC.DataBindings.Add(„Text“, filmotekaBindingSource, „predstavitel“, true, DataSourceUpdateMode.Never);
    ...
}
```



Označenie záložky

## Zviazanie objektu NamedRange s údajom z databázy v programe Excel

Objekt Range chápeme ako určitú spojitú oblasť dokumentu ohraničenú v prípade tabuľky Excelu pozíciou prvej a poslednej bunky. Kód nasledujúceho príkladu umiestni do bunky A1 údaj z databázy.

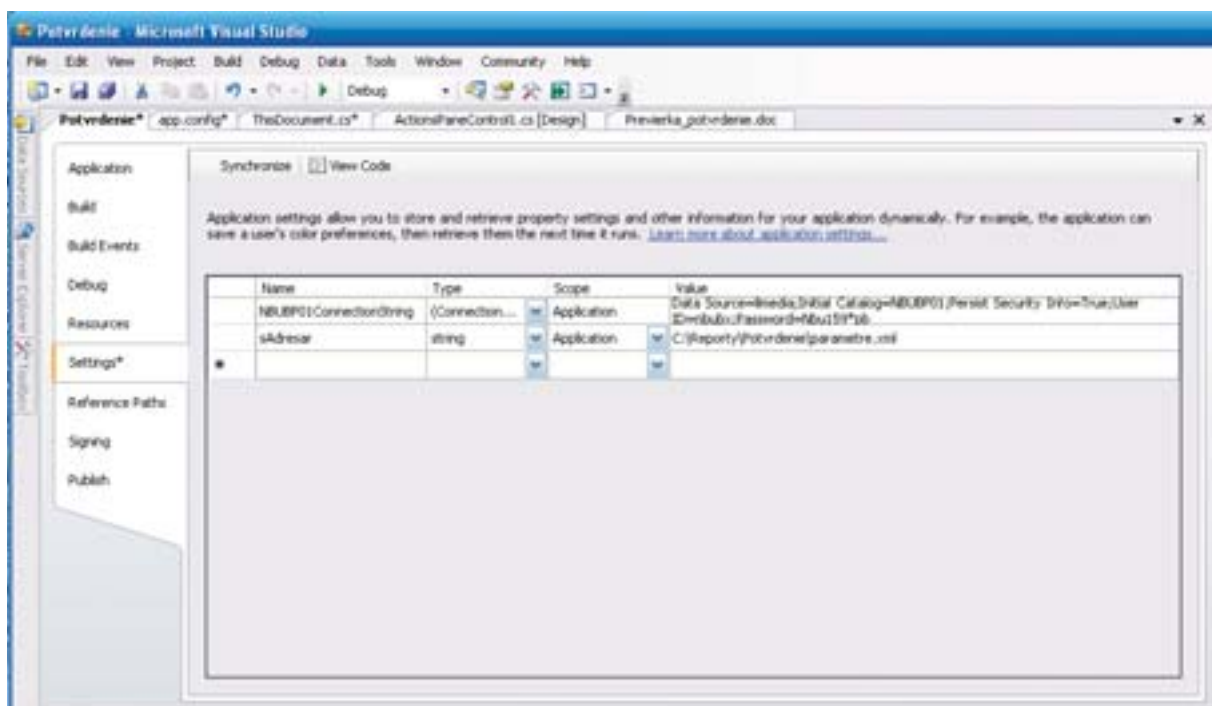
```
private void btAkcia_Click(object sender, EventArgs e)
{
    Microsoft.Office.Tools.Excel.NamedRange NamedRange1;
    NamedRange1 = this.Controls.AddNamedRange(Globals.Hárok1.Range[„A1“, missing], „NamedRange1“);
    NamedRange1.DataBindings.Add(„Value2“, filmotekaBindingSource,„predstavitel“);
}
```

## KAPITOLA 7: PRÁCA S OBJEKTMI A DOKUMENTMI

Podobne ako programy z balíka Microsoft Office sú určené na prácu s jedným alebo viacerými dokumentmi, aj „Smart“ aplikácia vytvorená pomocou VSTO dokáže pracovať s viacerými dokumentami, aj keď podobne ako programy Office nie naraz.

### Ukladanie konfiguračných parametrov

Okrem údajov, ktoré si aplikácia zhromažďuje a udržiava počas svojho behu, sú pre jej úspešné spustenie a fungovanie potrebné aj určité konfiguračné parametre. Typickým príkladom konfiguračných údajov aplikácie sú reťazce parametrov pre pripojenie sa aplikácie k databázovému serveru. Pre ukladanie globálnych parametrov nastavenia aplikácie môžeme s výhodou využiť súbor app.config. Tento súbor môžeme editovať buď priamo, alebo prostredníctvom formulára pre nastavenie parametrov aplikácie, presnejšie v záložke „Settings“. V prvom prípade editujeme textovú podobu XML dokumentu. V prípade využitia formulára sú jednotlivé parametre usporiadané v riadkoch tabuľky, kde ich môžeme vo Visual Studiu editovať.



Tabuľka parametrov aplikácie

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    ...
  </configSections>
  <connectionStrings>
    <add name="Potvrdenie.Properties.Settings.REPConnectionString"
      connectionString="Data Source=IImedia;Initial Catalog=REP;Persist Security Info=True;User
ID=user1;Password=Pass@word1"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
  <applicationSettings>
    <Potvrdenie.Properties.Settings>
      <setting name="sAdresar" serializeAs="String">
        <value>C:\Reporty\Potvrdenie\parametre.xml</value>
      </setting>
    </Potvrdenie.Properties.Settings>
  </applicationSettings>
</configuration>
```

```
</setting>
</Potvrdenie.Properties.Settings>
</applicationSettings>
</configuration>
```

Do konfiguračného súboru app.config môžeme uložiť aj iné konfiguračné údaje, napríklad URL adresa webovej služby využívanej aplikáciou a podobne.

```
<applicationSettings>
  <Schvalovanie.Properties.Settings>
    <setting name="Schvalovanie_localhost_Service" serializeAs="String">
      <value>http://localhost/WSsec/service.asmx</value>
    </setting>
  </Schvalovanie.Properties.Settings>
</applicationSettings>
```

Parameter v aplikácii zo sekcie <applicationSettings> prečítame nasledovne (názov aplikácie je Potvrdenie)

```
string sParamDir;
sParamDir = Potvrdenie.Properties.Settings.Default.sAdresar;
```

Reťazce zo sekcie ConnectionStrings v aplikácii čítame pomocou konštrukcie

```
string sConn;
sConn = Potvrdenie.Properties.Settings.REPConnectionString;
```

## Excel - práca s dokumentami

U Excelu rozlišujeme prácu s dokumentami na jednak úrovni celého zošitu - Workbooku a taktiež aj na nižšej úrovni granularity – na úrovni i jednotlivých hárkov.

### Práca s dokumentom na úrovni Workbook

Fungovanie rutín na úrovni Workbook si ukážeme na jednoduchom príklade, kedy pozatvárame všetky dcérske hárky.

#### Visual Basic

```
Public Sub AkciaWorkbooku()
    ThisApplication.Workbooks.Close()
End Sub
```

#### C#

```
public void AkciaWorkbooku()
{
    ThisApplication.Workbooks.Close();
}
```

Znovu zdôrazňujeme, že tento kód je spustený na úrovni objektu Workbook, teda v súbore „ThisWorkbook.vb“ či „ThisWorkbook.cs“. Na úrovni kódu hárkov je potrebné pristupovať ku kódu na úrovni Workbooku cez globálne objekty.

### Visual Basic

```
Private Sub btAkcia_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btAkcia.Click  
    Globals.ThisWorkbook.Close()  
End Sub
```

### C#

```
private void btAkcia_Click(object sender, EventArgs e)  
{  
    Globals.ThisWorkbook.Close(false, missing, missing);  
}
```

Úmyselne sme volili najskôr príklad pre zatvorenie hárkov nakoľko nevyžaduje nijaké parametre, a predsa len sa jednoduchšie niečo zruší ako vytvorí. Následne ukážeme príklad pre otvorenie hárku. Tu sa musíme vysporiadať s parametrami. U aplikácie v programovacom jazyku C# musíme vymenovať všetky deklarované parametre, pričom nevyužívané parametre nahradíme deklaráciou „missing“.

### Visual Basic

```
Dim wb As Excel.Workbook = Me.Application.Workbooks.Open(„C:\kluce.xls“)
```

### C#

```
Excel.Workbook wb = this.Application.Workbooks.Open(“C:\kluce.xls”,  
    missing, missing, missing, missing, missing, missing, missing,  
    missing, missing, missing, missing, missing, missing, missing);
```

## Práca s dokumentom na úrovni pracovných hárkov

Z úrovne dokumentov môžeme prejsť na nižšiu hierarchickú úroveň jednotlivých hárkov. Pokúsme sa najskôr vybrať ako aktívny hárok „Hárok 3“.

### Visual Basic

```
Globals.Hárok3.Select()
```

### C#

```
Globals.Hárok3.Select(missing);
```

Predchádzajúca konštrukcia je možná v prípade ak poznáme názov príslušného hárku. Ak poznáme len poradie hárku môžeme aplikovať kód v modifikovanom tvare.

### Visual Basic

```
CType(Me.Application.ActiveWorkbook.Sheets(2), Excel.Worksheet).Select()
```

### C#

```
((Excel.Worksheet)this.Application.ActiveWorkbook.Sheets[2]).Select(missing);
```

Jednotlivé hárky môžu byť viditeľné, alebo skryté nastavujeme to pomocou parametrov

### Visual Basic

```
Globals.Hárok1.Visible = Excel.XISheetVisibility.xlSheetHidden  
Globals.Hárok1.Visible = Excel.XISheetVisibility.xlSheetVisible
```

### C#

```
Globals.Hárok2.Visible = Excel.XISheetVisibility.xlSheetHidden;  
Globals.Hárok2.Visible = Excel.XISheetVisibility.xlSheetVisible;
```

Ak potrebujeme operatívne zistiť napríklad názvy jednotlivých hárkov, použijeme k tomu programovú konštrukciu

### Visual Basic

```
Dim index As Integer = 0
```

```
Dim rng As Excel.Range = Globals.Hárok1.Range(„A1“)  
For Each displayWorksheet As Excel.Worksheet In Globals.ThisWorkbook.Worksheets  
    rng.Offset(index, 0).Value2 = displayWorksheet.Name  
    index += 1  
Next displayWorksheet
```

### C#

```
int index = 0;
```

```
Microsoft.Office.Tools.Excel.NamedRange NamedRange1 =  
    Globals.Hárok1.Controls.AddNamedRange(  
        Globals.Hárok1.Range[„A1“, missing], „NamedRange1“);  
  
foreach (Excel.Worksheet displayWorksheet in Globals.ThisWorkbook.Worksheets)  
{  
    NamedRange1.Offset[index, 0].Value2 = displayWorksheet.Name;  
    index++;  
}
```

## Word - práca s dokumentami

Na rozdiel od Excelu kde sa dokument člení na niekoľko hárkov, u dokumenty programu Word sú spravidla tvorené jedným súborom.

### Ukladanie dokumentov do súborov

Dokument uložíme do súboru pomocou metódy SaveAs. Táto metóda má 15 parametrov, z ktorých využijeme len niektoré, minimálne jeden povinný a to názov súboru. Pomerne často sa využíva aj parameter ReadOnly.

```

Document SaveAs(
    [In] ref object FileName,
    [In, Optional] ref object ConfirmConversions,
    [In, Optional] ref object ReadOnly,
    [In, Optional] ref object AddToRecentFiles,
    [In, Optional] ref object PasswordDocument,
    [In, Optional] ref object PasswordTemplate,
    [In, Optional] ref object Revert,
    [In, Optional] ref object WritePasswordDocument,
    [In, Optional] ref object WritePasswordTemplate,
    [In, Optional] ref object Format,
    [In, Optional] ref object Encoding,
    [In, Optional] ref object Visible,
    [In, Optional] ref object OpenAndRepair,
    [In, Optional] ref object DocumentDirection,
    [In, Optional] ref object NoEncodingDialog,
    [In, Optional] ref object XMLTransform
);

```

Pre ostatné, voliteľné parametre budeme musieť v programovacom jazyku C# odovzdať referenciu na objekt Nic (znamená to doslova a do písmena nič), čiže anglicky Missing Value.

```
object Nic = System.Reflection.Missing.Value;
```

Kompletný program ako obslužná procedúra udalosti zatlačenia nejakého tlačidla by mohla byť v tvare

```

private void btUlozDokument_Click(object sender, EventArgs e)
{
    object Nic = System.Reflection.Missing.Value;
    object Subor = (object)@"c:\dokument.doc";

    this.SaveAs(ref Subor,
        ref Nic, ref Nic, ref Nic, ref Nic,
        ref Nic, ref Nic, ref Nic, ref Nic,
        ref Nic, ref Nic, ref Nic, ref Nic,
        ref Nic, ref Nic, ref Nic);
}

```

Namiesto vytvárania objektu „chýbajúcej hodnoty“ môžeme použiť referenciu na objekt „missing“

```

private void btUlozDokument_Click(object sender, EventArgs e)
{
    object Subor = (object)@"c:\dokument.doc";
    this.SaveAs(ref Subor,
        ref missing, ref missing, ref missing, ref missing,
        ref missing, ref missing, ref missing, ref missing,
        ref missing, ref missing, ref missing, ref missing,
        ref missing, ref missing, ref missing);
}

```

Vo Visual Basicu sa tento kód zredukuje na jeden riadok

```
Me.SaveAs(,c:\dokument.doc")
```

Na tomto príklade je zrejmy nielen spôsob ako po ukončení práce s aktívnym dokumentom tento dokument uložiť, ale aj to, že kód v programovacom jazyku je v tomto prípade, teda pre otvorenie dokumentu oveľa stručnejší v programovacom jazyku Visual Basic.



## Otvorenie dokumentu zo súboru

Povaha wordu ako textového editora ho predurčuje pre prácu s textovými dokumentami v súboroch. Aj keď VSTO aplikácia sa vlastne spúšťa načítaním wordového dokumentu, nastane možno prípad, kedy VSTO aplikácia bude potrebovať pracovať s viacerými dokumentami, napríklad prekopírovať časť obsahu z novootvoreného dokumentu. Metóda pre otvorenie dokumentu môže mať tieto parametre:

```
Document Open(  
    [In] ref object FileName,  
    [In, Optional] ref object ConfirmConversions,  
    [In, Optional] ref object ReadOnly,  
    [In, Optional] ref object AddToRecentFiles,  
    [In, Optional] ref object PasswordDocument,  
    [In, Optional] ref object PasswordTemplate,  
    [In, Optional] ref object Revert,  
    [In, Optional] ref object WritePasswordDocument,  
    [In, Optional] ref object WritePasswordTemplate,  
    [In, Optional] ref object Format,  
    [In, Optional] ref object Encoding,  
    [In, Optional] ref object Visible,  
    [In, Optional] ref object OpenAndRepair,  
    [In, Optional] ref object DocumentDirection,  
    [In, Optional] ref object NoEncodingDialog,  
    [In, Optional] ref object XMLTransform  
);
```

Kompletný kód v jazyku C# pre načítanie dokumentu v metóde pre obsluhu udalosti zatlačenia tlačidla bude.

```
private void btOtvorDokument_Click(object sender, EventArgs e)  
{  
    object Subor = (object)@"c:\Stopar.doc";  
    this.Application.Documents.Open(ref Subor,  
        ref missing, ref missing, ref missing, ref missing,  
        ref missing, ref missing, ref missing, ref missing,  
        ref missing, ref missing, ref missing, ref missing,  
        ref missing, ref missing, ref missing);  
}
```

## Ukladanie dokumentov do databázy

Pre ukladanie obsahu dokumentov do databáz sa hodia viaceré dátové typy, V novej verzii databázového SQL Server 2005 bol zavedený nový dátový typ varbinary(max) pre ukladanie rozsiahlych binárnych dokumentov. Tento dátový typ prelamuje dvojgigabajtovú hranicu pre ukladanie veľkých blokov údajov. Umožňujú uložiť až  $2^{31}-1$  bajtov. Ukladanie binárnych dokumentov do databázy sa v praxi pomerne často používa, napríklad aj pre ukladanie dokumentov vytvorených vo formátoch rôznych kancelárskych programov, napríklad textového editora Microsoft Word, alebo tabuľkového programu Excel.

Pre ukladanie dokumentov použijeme buď nejakú vhodnú už existujúcu databázu, alebo vytvoríme novú databázu. Pre vytvorenie novej databázy môžeme zadať cez konzolovú aplikáciu príkaz.

```
CREATE DATABASE test
```

Vo vybranej databáze vytvoríme tabuľku pre ukladanie dokumentov. Takáto tabuľka musí mať minimálne dva stĺpce, názov dokumentu a binárny obsah dokumentu. Ak zaistíme, že názvy sú unikátne, môže byť stĺpec s názvom dokumentu aj primárnym kľúčom. Dva stĺpce sú však len teoretickým minimom. Do takejto tabuľky môžeme ukladať len dokumenty jedného typu, napríklad len

dokumenty textového editora Word alebo len dokumenty programu Excel. Ak by sme chceli do tabuľky ukladať viac typov dokumentov, museli by sme ich rozlíšiť podľa typu, museli by sme teda pridať ďalší stĺpec, do ktorého by sme zadávali napríklad príponu úboru (DOC v prípade dokumentu textového editora Word, XLS pre dokumenty programu Excel). Spravidla ešte do tabuľky dokumentov pridávame dátum kedy bol dokument vytvorený, dátum, kedy bol dokument naposledy modifikovaný, prípadne príznaky, kto s dokumentom pracoval, či je už schválený alebo nie a podobne. Pre naše účely vystačíme s takouto tabuľkou.

```
CREATE TABLE Dokumenty
(
  ID int identity(1,1) not for replication primary key,
  NazovSuboru nvarchar(60),
  TypSuboru nvarchar(6),
  Dokument varbinary(max),
  DatumSpracovania datetime default GETDATE()
)
```

Dokument bude fyzicky uložený v stĺpci typu VARBINARY(MAX). Dva stĺpce typu NVARCHAR(60) a NVARCHAR(6) budú obsahovať údaje o názve súboru a jeho type, presnejšie povedané, prípone. Skôr než začneme pracovať s dokumentami vo VSTO aplikáciách, jeden dokument uložíme do databázy priamo, príkazom cez klientskú konzolovú aplikáciu. Dokument v súbore otvoríme pomocou funkcie OPENROWSET, pričom stream binárnych údajov vytvoríme pomocou funkcie BULK.

```
INSERT INTO Dokumenty(NazovSuboru, TypSuboru, Dokument)
SELECT ‚LoremIpsum.doc‘ AS NazovSuboru, ‚.doc‘ AS TypSuboru, * FROM
OPENROWSET(BULK N‘C:\LoremIpsum.doc‘, SINGLE_BLOB) AS Dokument;
```

## Výber dokumentu z databázy a jeho uloženie do súboru

Zatiaľ, čo pre uloženie dokumentu zo súboru do databázy sme vystačili s SQL príkazom obsahujúcim funkciu BULK, inverzná funkcia, k funkcii BULK, pomocou ktorej by sme vybrali dokument z databázy, uložili ho do súboru a otvorili ho nie je k dispozícii, takže túto problematiku musíme riešiť pomocou programového kódu aplikácie. V nasledujúcej ukážke je príklad procedúry pre načítanie dokumentu so zadaným identifikačným číslom z databázy a jeho uloženie do súboru.

```
public void DokumentDoSuboru(int index, string sNazov)
{
  SqlConnection conn = null;
  SqlCommand cmd = null;
  SqlParameter param = null;
  SqlDataReader sdr = null;
  FileStream fs = null;
  BinaryWriter bw = null;
  long blob;
  long startIndex = 0;
  byte[] outBuffer = new byte[255];
  string sConn = „server=llmedia;database=test;integrated security=true“;

  try
  {
    conn = new SqlConnection(sConn);
    cmd = new SqlCommand(„SELECT Dokument FROM Dokumenty WHERE ID=@ID“, conn);
    param = new SqlParameter(„@ID“, SqlDbType.Int);
    param.Value = index;
    cmd.Parameters.Add(param);
    conn.Open();
    sdr = cmd.ExecuteReader(CommandBehavior.SequentialAccess);
```

```

while (sdr.Read())
{
    fs = new FileStream(sNazov, FileMode.OpenOrCreate, FileAccess.Write);
    bw = new BinaryWriter(fs);
    blob = 255;
    while (blob == 255)
    {
        blob = sdr.GetBytes(0, startIndex, outBuffer, 0, 255);
        bw.Write(outBuffer);
        bw.Flush();
        startIndex += 255;
    }
    bw.Close();
    fs.Close();
    sdr.Close();
    conn.Close();
}
catch (SqlException ex)
{
    MessageBox.Show(„SQL Exception: „ + ex.Message);
}
}

```

## Výber dokumentu zo súboru a jeho uloženie do databázy

Variantu uloženia wordového dokumentu zo súboru do databázy pomocou príkazu BULK sme ukázali v jednej z predchádzajúcich statí. Ten istý cieľ dosiahneme aj pomocou aplikačného kódu.

```

private void btDokumentToDb_Click(object sender, EventArgs e)
{
    SqlConnection conn = null;
    SqlCommand cmd = null;
    SqlParameter param = null;
    FileStream fs = null;
    const string sConn = „server=\\media;database=test;integrated security=true“;
    try
    {
        conn = new SqlConnection(sConn);
        cmd = new SqlCommand(„INSERT INTO Dokumenty(NazovSuboru, Dokument)VALUES (,Stopar.doc', @
Dokument)“, conn);
        fs = new FileStream(„c:\Stopar.doc“, FileMode.Open, FileAccess.Read);
        Byte[] blob = new Byte[fs.Length];
        fs.Read(blob, 0, blob.Length);
        fs.Close();

        param = new SqlParameter(„@Dokument“, SqlDbType.VarBinary, blob.Length, ParameterDirection.Input, false,
0, 0, null, DataRowVersion.Current, blob);
        cmd.Parameters.Add(param);
        conn.Open();
        cmd.ExecuteNonQuery();
    }
    catch (SqlException ex)
    {
        MessageBox.Show(„SQL Exception: „ + ex.Message);
    }
    catch (Exception ex) { MessageBox.Show(„Exception: „ + ex.Message); }
}

```

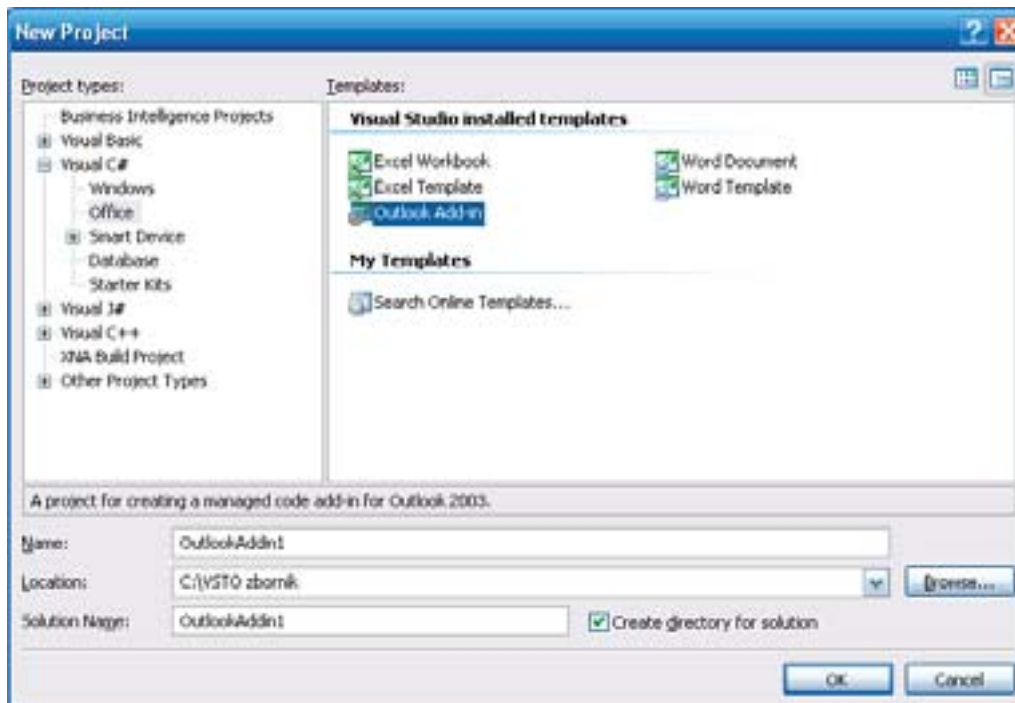
## KAPITOLA 8: OUTLOOK

Filozofia inštalácie a fungovanie aplikácií pre Outlook je podstatne odlišná ako u aplikácií pre Word a Excel. Zatiaľ čo u Wordu a Excelu sa kód aplikácie vo forme dll pripojí k dokumentu, takže k rôznym dokumentom môžu byť pripojené rôzne aplikácie, vyvinuté VSTO doplnky sa pripájajú priamo k programu Outlook, takže môžu byť aktívne vo všetkých outlookom podporovaných režimoch. Je to logické, pretože aj napriek tomu, že program Outlook dokáže pracovať s niektorými druhmi dokumentov, napríklad vizitky (súbor s príponou VCF), no nie je to jeho typický režim práce a zjednodušene povedané, doplnok by vlastne nebolo k akému dokumentu pripojiť.

Druhou otázkou je filozofia zakomponovania funkcionality, ovládacích prvkov a zobrazenia výsledkov aktivít zobrazeného kódu. U Wordu a Excelu bolo možné ovládacie prvky rozmiestniť buď priamo na ploche konkrétneho dokumentu, alebo na vhodne umiestnený panel ovládacích prvkov.

### Vytvorenie nového VSTO projektu doplnku pre Outlook

V dialógu pre vytvorenie nového projektu vyberieme v požadovanom programovacom jazyku šablónu „Outlook Add-in“.



Dialóg pre vytvorenie nového VSTO projektu typu Outlook Add-in

Po vytvorení projektu vznikne prázdny zdrojový kód v príslušnom programovacom jazyku, ktorý obsahuje telá dvoch procedúr „Startup“ a „Shutdown“.

V jazyku **C#** je úvodný kód vytvorený sprievodcom takýto

```
using System;  
using System.Windows.Forms;  
using Microsoft.VisualStudio.Tools.Applications.Runtime;  
using Outlook = Microsoft.Office.Interop.Outlook;  
using Office = Microsoft.Office.Core;
```

```
namespace OutlookAddin1  
{  
    public partial class ThisApplication  
    {
```

```

private void ThisApplication_Startup(object sender, System.EventArgs e)
{
}

private void ThisApplication_Shutdown(object sender, System.EventArgs e)
{
}

#region VSTO generated code

}
}

```

V nezobrazenom odstavci *#region VSTO generated code* je kód, ktorý priradí rutiny pre „Startup“ a „Shutdown“ príslušným udalostiam.

```

private void InternalStartup()
{
    this.Startup += new System.EventHandler(ThisApplication_Startup);
    this.Shutdown += new System.EventHandler(ThisApplication_Shutdown);
}

```

V jazyku **Visual Basic**

```

public class ThisApplication

```

```

    Private Sub ThisApplication_Startup(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Startup
    End Sub

```

```

    Private Sub ThisApplication_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Shutdown
    End Sub

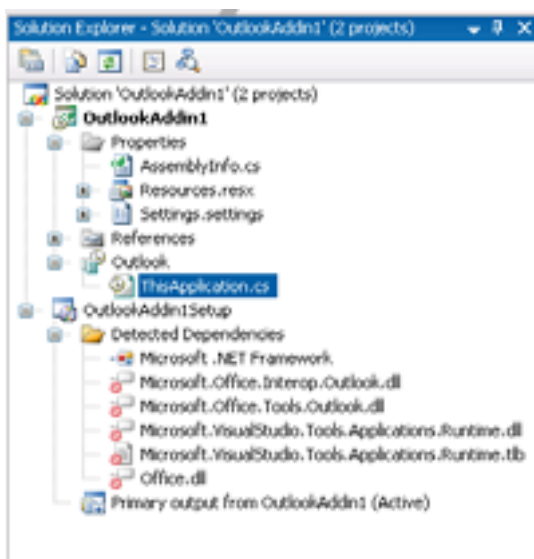
```

```

End class

```

Pri prvom založení prázdnej aplikácie určite presne nezistíme, na akom princípe funguje, ani to aký je účel jednotlivých súborov tvoriacich projekt. Pri pohľade na stromovú štruktúru súborov z ktorých projekt pozostáva nám stačí poznať, že kód budeme písať do súboru *ThisApplication.cs*, prípadne ešte skutočnosť, že súčasťou projektu sú aj inštaláčne súbory a dll.



Projekt tvoria dve šablóny

Na záver zoznamovania sa s prvým prázdny projektom si všimnite, že na rozdiel od VSTO aplikácií pre Word a Excel, súbor ThisApplication.cs obsahuje len kód, to znamená že tento dokument nie je možné prepnúť do vizuálneho návrhového režimu. Ak si preštudujeme ďalšie príklady, bude zrejmé prečo je tomu tak. Dialógové okná obsahujúce ovládacie prvky samozrejme do vizuálneho návrhového módu prepnúť môžeme.

## Zistenie počtu správ v schránke elektronickej pošty

Na rozdiel od VSTO aplikácií pre Word a Excel, kde môžeme do textu, alebo do tabuľky vypisovať text a teda ako prvá ukážková VSTO aplikácia sa hodí aplikácia „Hello Word“, ktorá na obrazovku vypíše nejaký text, alebo obsahy nejakých premenných, primárne určenie Outlooku je trochu iné. Preto ako námet pre prvú aplikáciu použijeme výpis počtu správ v schránke elektronickej pošty.

### C#

```
private void ThisApplication_Startup(object sender, System.EventArgs e)
{
    //referencia na schranku elektronickej pošty
    Outlook.MAPIFolder Schranka;
    Schranka =
        this.Session.GetDefaultFolder(Outlook.OIDefaultFolders.olFolderInbox);

    //zobraz pocet sprav
    MessageBox.Show("Mate " + Schranka.Items.Count + " sprav v schranke.");
}

```

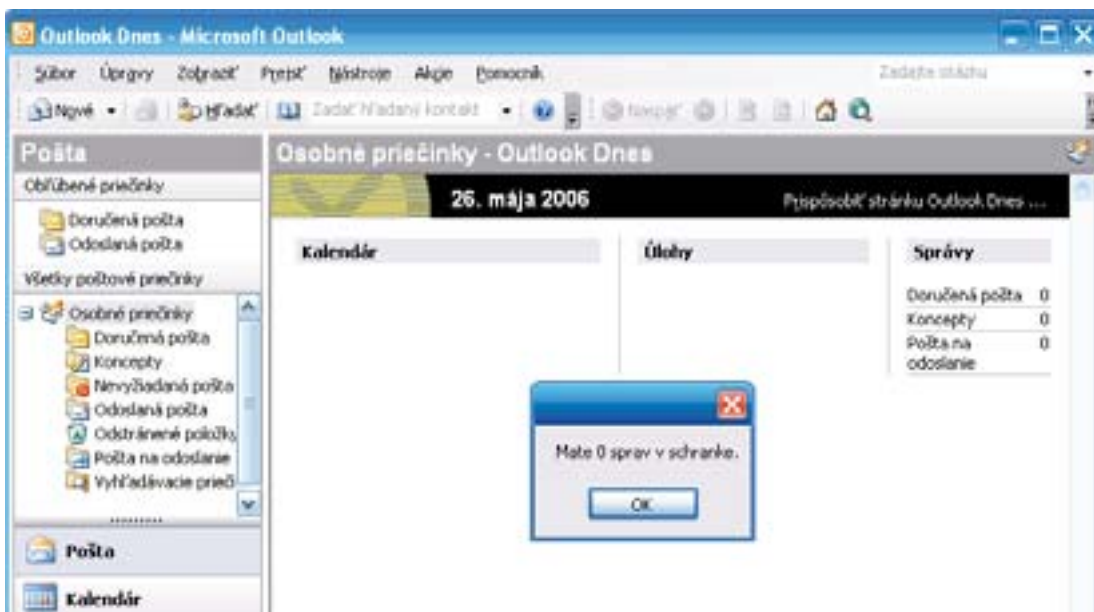
### Visual Basic

```
Private Sub ThisApplication_Startup(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Startup
    'referencia na schranku elektronickej pošty
    Dim Schranka As Outlook.MAPIFolder
    Schranka = Me.Session.GetDefaultFolder(Outlook.OIDefaultFolders.olFolderInbox)

    'zobraz pocet sprav
    MessageBox.Show("Mate " & Schranka.Items.Count & " sprav v schranke.")

End Sub

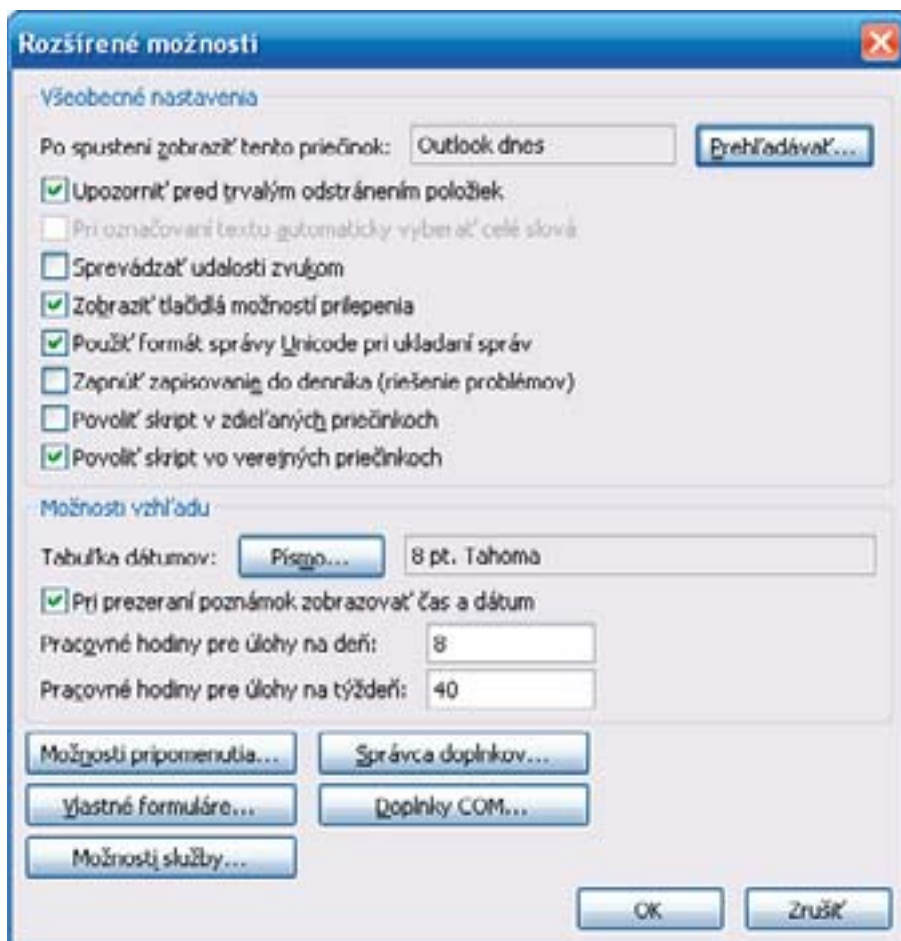
```



Aplikácia pre výpis počtu správ v schránke elektronickej pošty

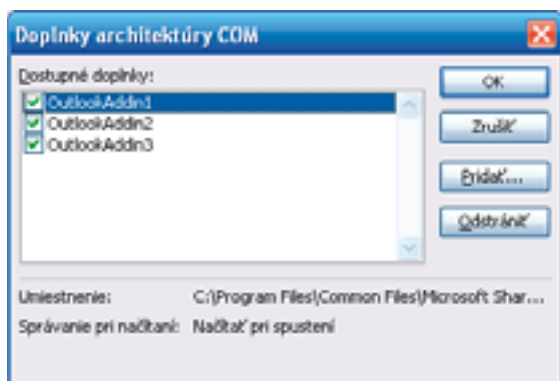
## Povolenie a zakázanie doplnku pre Outlook

Po úspešnom odladení a vyskúšaní prvého doplnku pre Outlook vo vývojovom prostredí, môžeme ladenie ukončiť, vývojové prostredie zatvoriť a znovu spustiť program Outlook. Zistíme, že nami vyvinutý doplnok je stále aktívny, to znamená, že po spustení sa vypíše dialógové okno s počtom správ v schránke. Ak chceme doplnok zakázať, využijeme priamo v programe Outlook menu „Tools“, „Options“, vyberieme záložku „Other“, a nakoniec aktivujeme „Advanced Options“, prípadne v jazyku nášho kmeňa „Nástroje“, „Možnosti“, záložka „Ďalšie“, a tlačidlo „Rozšírené možnosti“.



Dialóg pre nastavenie rozšírených možností

V tomto dialógu využijeme tlačidlo „COM Add-in“ alebo po našom „Doplnky COM“. V dialógu pre správu doplnkov architektúry COM môžeme nami vyvinutý doplnok pre program Outlook povoliť, zakázať, odobrať, ale aj pridať.



Povolenie alebo zakázanie doplnkov



## Vytvorenie a odoslanie mailu

Určitou analógiou textového wordového dokumentu, prípadne excelového hárku sú u programu Outlook správy elektronickej pošty. Preto námetom ďalšieho príkladu bude vytvorenie a poslanie správy elektronickej pošty. Postup úkonov, ktoré vykoná náš kód bude určitou analógiou postupu, ktorý vykonávame, keď posielame pomocou Outlooku e-mail ručne.

- prepnutie programu Outlook do režimu vytvorenia nového e-mailu
- vyplnenie parametrov- adresy adresáta, predmetu a textu správy
- odoslanie správy

### C#

```
private void ThisApplication_Startup(object sender, System.EventArgs e)
{
    // Novy objekt e-mail item
    Outlook.Mailltem mailltem =
        (Outlook.Mailltem)this.CreateItem(Outlook.OlltemType.olMailltem);

    //nalezitosti mailu
    mailltem.Subject = "Hello World!"; // Predmet
    mailltem.To = "llacko@pcrevue.sk"; //adresa
    mailltem.Body = "Hello World"; // Text
    mailltem.Display(false); //zobraz mail

    // Ponuka na odoslanie
    if (MessageBox.Show("Chcete mail odoslat?", "Pokus", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == DialogResult.Yes)
    {
        ((Outlook._Mailltem)mailltem).Send();
    }
    else
    {
        ((Outlook._Mailltem)mailltem).Close(Outlook.OllnspectorClose.olDiscard);
    }
}
```

### Visual Basic

```
Private Sub ThisApplication_Startup(ByVal sender As _
    Object, ByVal e As System.EventArgs) Handles _
    Me.Startup

    ' Novy objekt e-mail item
    Dim mailltem As Outlook.Mailltem = _
        Me.CreateItem(Outlook.OlltemType.olMailltem)

    ' nalezitosti mailu
    mailltem.Subject = "Hello World!"
    mailltem.To = "llacko@pcrevue.sk"
    mailltem.Body = "Hello World."
    mailltem.Display(False)

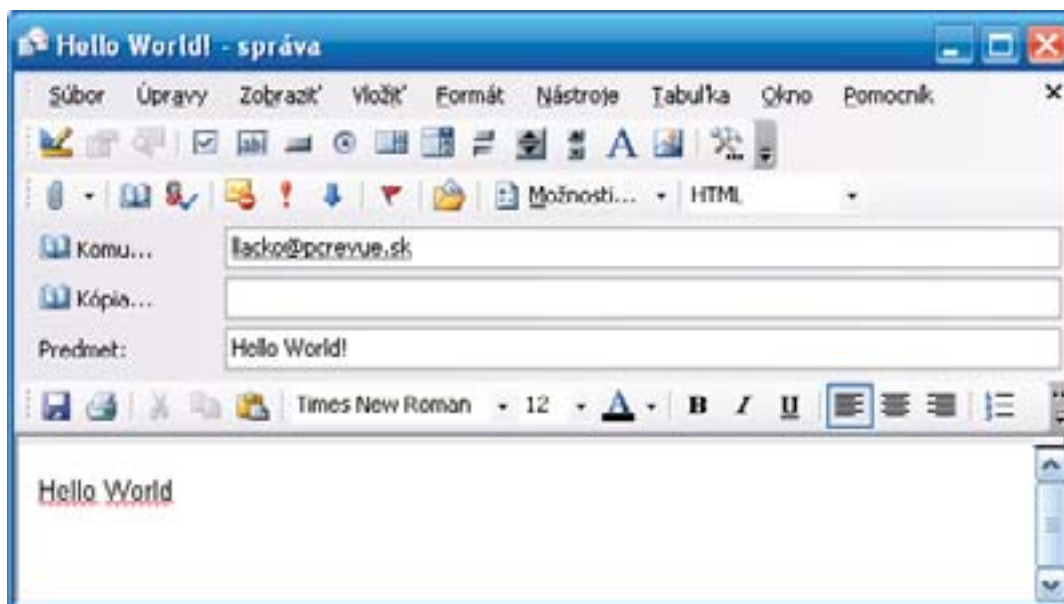
    ' Ponuka na odoslanie
    If (MessageBox.Show("Chcete mail odoslat?", _
        "HandsOnLab.Lab", _
```

```

    MessageBoxButtons.YesNo, MessageBoxIcon.Question) _
    = DialogResult.Yes) Then
    mailItem.Send()
Else
    mailItem.Close(Outlook.OlInspectorClose.olDiscard)
End If
End Sub

```

K ovládacím prvkom pre aktivovanie kódu sa prepracujeme až v nasledujúcom príklade, preto ako udalosť pre aktivovanie vytvorenia a odoslania správy elektronickej pošty využijeme spustenie aplikácie Outlook.



Aplikácia pre vytvorenie a odoslanie mailu

## Pridanie nového kontaktu

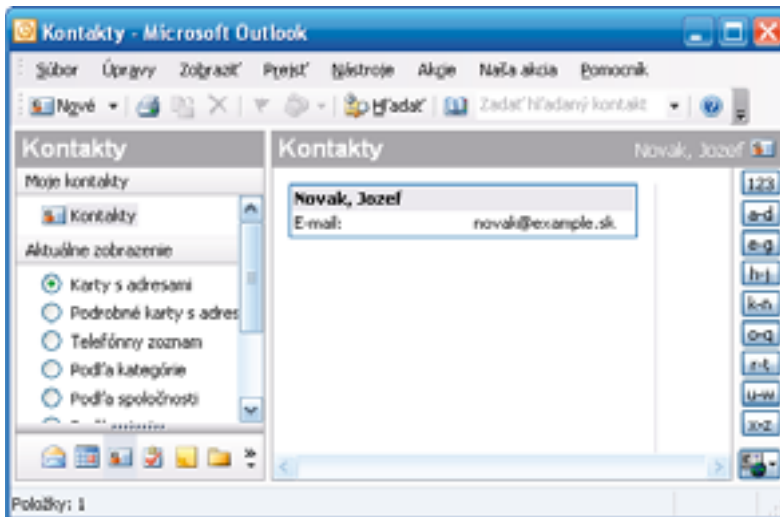
Vytvorením a odoslaním mailu z aplikácie Outlook pomocou kódu vo VSTO doplnku sme čiastočne naznačili využitie jednej z hlavných funkcionalít aplikácie Outlook – mailového klienta. Podobne môžeme pomocou kódu pracovať s kontaktami. Ukážeme príklad kódu pre vytvorenie nového kontaktu. Najskôr zadáme len nevyhnutné parametre.

### C#

```

private void ThisApplication_Startup(object sender, System.EventArgs e)
{
    Outlook.ContactItem contact = (Outlook.ContactItem)this.CreateItem(Outlook.OlItemType.olContactItem);
    contact.FirstName = "Jozef";
    contact.LastName = "Novak";
    contact.Email1Address = "novak@example.sk";
    contact.Categories = "Hobby";
    contact.Save();
}

```



Pridanie nového kontaktu

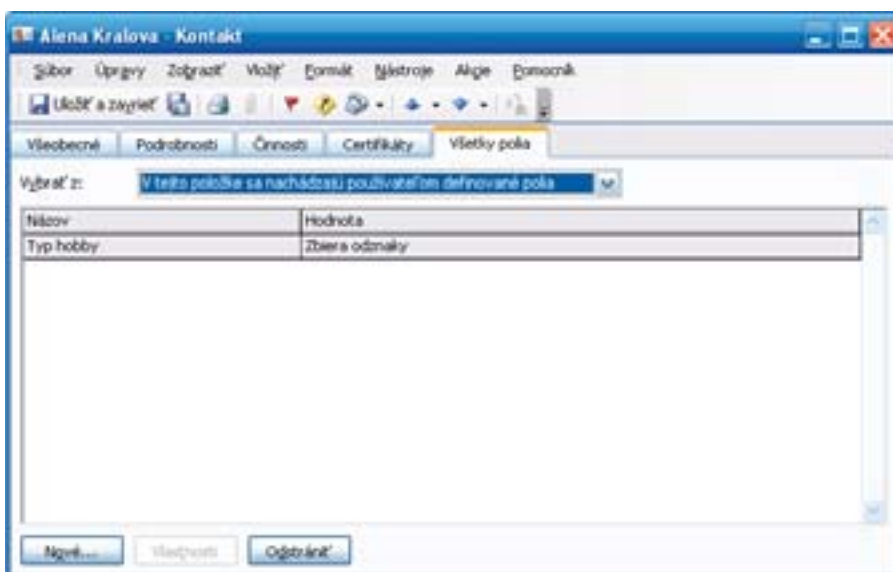
Ku každému kontaktu je možné pridať ľubovoľné množstvo používateľom definovaných atribútov. V ďalšom príklade pridáme atribút „Typ Hobby“. Typ atribútu bude textový reťazec. Zároveň v kóde nastavíme aj hodnotu atribútu.

### C#

```
private void ThisApplication_Startup(object sender, System.EventArgs e)
{
    Outlook.ContactItem contact = (Outlook.ContactItem)this.CreateItem(Outlook.OlltemType.olContactItem);
    contact.FirstName = „Alena“;
    contact.LastName = „Kralova“;
    contact.Email1Address = „kralova@example.sk“;
    Outlook.UserProperty uProperty;
    uProperty = contact.UserProperties.Add(„Typ hobby“,
    Outlook.OlUserPropertyType.olText, Type.Missing, Type.Missing);
    uProperty.Value = „Zbiera odznaky“;
    contact.Categories = „Hobby“;
    contact.Save();
}

```

Ak následne spustíme Outlook a prezrieme si novo pridaný kontakt. Používateľom definované vlastnosti budú u príslušného kontaktu zobrazené v záložke „Všetky polia“.



Zobrazenie používateľom definovaných vlastností

## Vytvorenie novej úlohy

Dôležitou súčasťou Outlooku je aj správa úloh (Tasks). Pridať novú úlohu by sme po doterajších skúsenostiach už zvládli, stačí nám poznať názvy objektov.

```
Outlook.TaskItem task =
(Outlook.TaskItem)this.CreateItem(Outlook.OllItemType.olTaskItem);
task.Subject = „Sledovat hobby aktivity „;
task.Body = „Umyt podlahu v klubovni.“;
task.Status = Outlook.OlTaskStatus.olTaskInProgress;
task.DueDate = DateTime.Today;
task.StartDate = DateTime.Today;
task.ReminderSet = true;

// pripomen o hodinu
task.ReminderTime = DateTime.Now + new TimeSpan(1, 0, 0);
task.Save();
```

## Vytvorenie úlohy pre kontakty z danej kategórie

Zaujímavejší príklad v spojitosti s úlohami bude pridanie úlohy pre každý kontakt z vybranej kategórie. Scenár bude nasledovný. Outlook je nainštalovaný na počítači klubovne. V kontaktoch máme niekoľko záznamov s kontaktami ľudí, ktorých máme zaradených do rôznych kategórií. Našou úlohou je vytvorenie aplikácie, ktorá pre každý kontakt z vybranej kategórie priradí nejakú konkrétnu úlohu. Jadrom aplikácie bude cyklus pre každý kontakt. Pre istotu skontrolujeme, či kontakt je platný a samozrejme, či patrí do kategórie, ktorá nás zaujme, v našom prípade „Hobby“.

Celý kód v jazyku C# (môžeme ho umiestniť do tela procedúry „startup“, alebo do obslužnej procedúry položky menu, vid' ďalej)

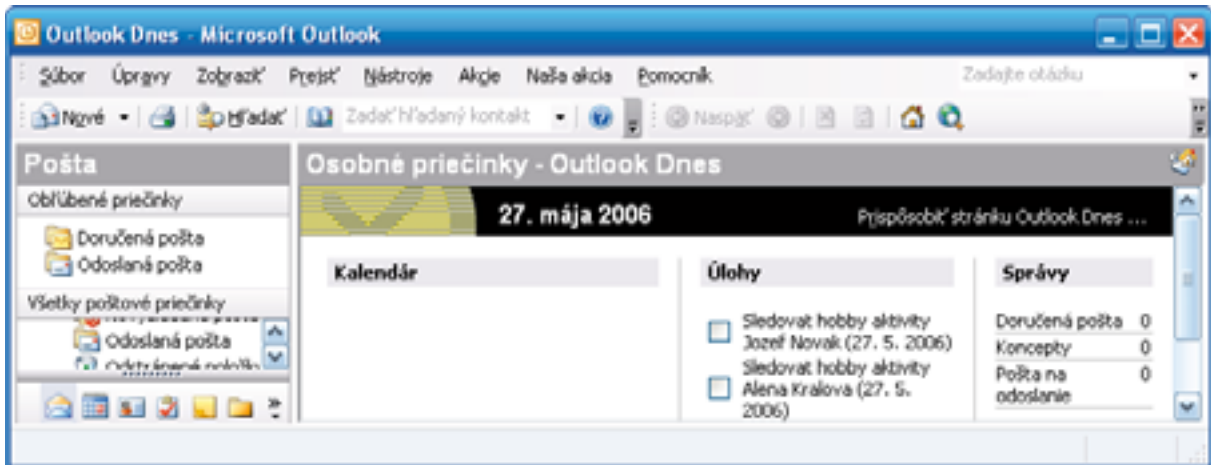
```
Outlook.MAPIFolder contacts =
this.ActiveExplorer().Session.GetDefaultFolder(Outlook.OlDefaultFolders.olFolderContacts);

//cyklus pre kontakty
for (int i = 1; i <= contacts.Items.Count; i++)
{
    Outlook.ContactItem contact =
        contacts.Items[i] as Outlook.ContactItem;

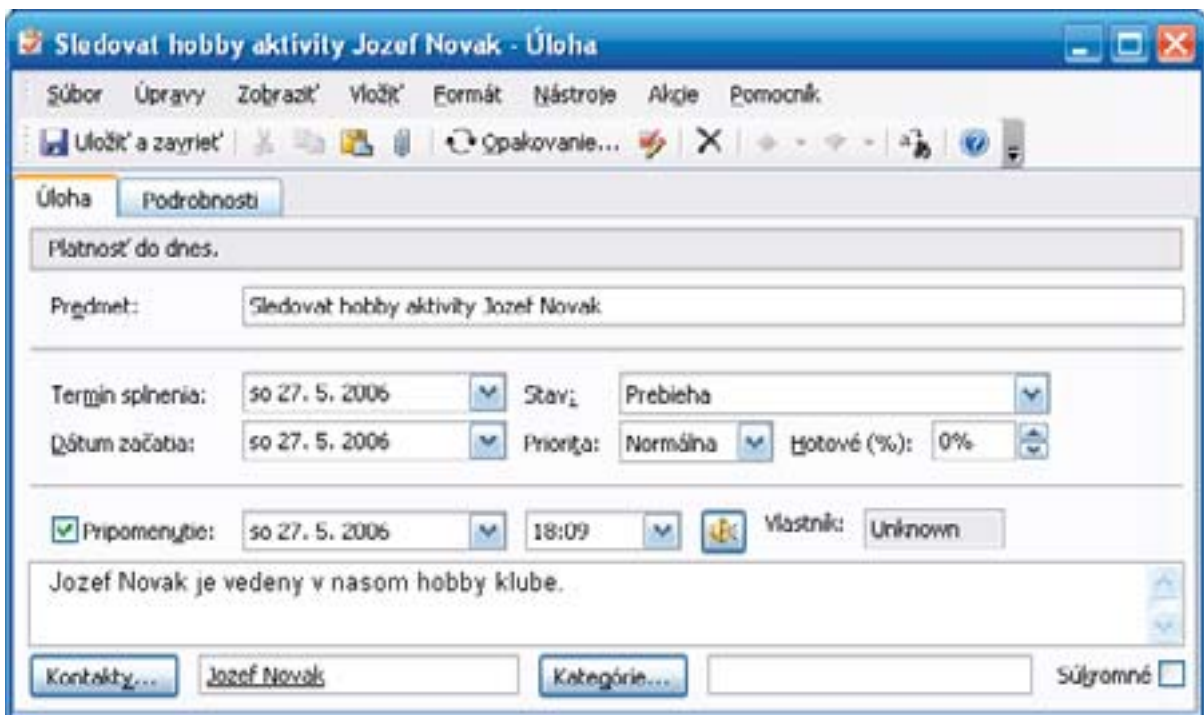
    if (contact != null)
    {
        if (contact.Categories == „Hobby“)
        {
            Outlook.TaskItem task =
                (Outlook.TaskItem)this.CreateItem(Outlook.OllItemType.olTaskItem);
            task.Subject = „Sledovat hobby aktivity „ + contact.FullName;
            task.Body = contact.FullName + „je vedeny v nasom hobby klube.“;
            task.Status = Outlook.OlTaskStatus.olTaskInProgress;
            task.DueDate = DateTime.Today;
            task.StartDate = DateTime.Today;
            task.ReminderSet = true;

            // pripomen o desat minut.
            task.ReminderTime = DateTime.Now + new TimeSpan(0, 10, 0);
            task.Links.Add(contact);
            task.Save();
        }
    }
}
```

O úspechu kódu sa môžeme presvedčiť, ak si necháme zobraziť zoznam úloh, prípadne si môžeme nechať zobraziť o vybranej úlohe detailné informácie



Zoznam úloh pridelených ku kontaktom

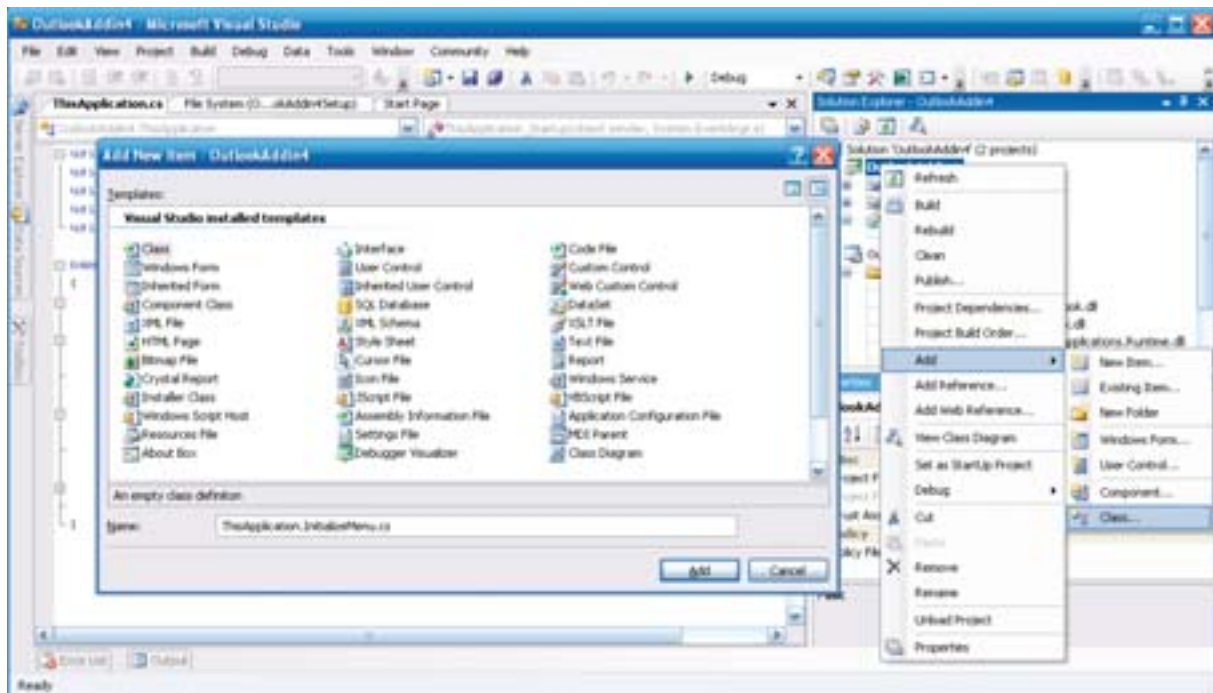


Detail úlohy

## Vytvorenie položky menu

Dosiaľ sme v obidvoch predchádzajúcich príkladoch, jednak pre výpis počtu nových správ elektronickej pošty a taktiež aj pre poslanie mailu využili ako udalosť pre aktiváciu nášho kódu spustenie programu Outlook. Kód bol zapúzdrený v tele obsluhy udalosti „Startup“. Ak chceme vykonať určitú akciu na pokyn používateľa musíme do doplnku zahrnúť nejaké aktivačné ovládacie prvky, napríklad menu.

Pre realizáciu novej položky hlavného menu vytvoríme novú triedu. Na obrázku je naznačený postup aktivácie kontextového menu „Add“, „Class“. Názov novej triedy zadáme „ThisApplication.InitializeMenu.cs“.



Pridanie triedy pre realizáciu menu

Sprievodca vytvorením novej triedy vygeneruje jej prázdny kód

### C#

```
using System;  
using System.Collections.Generic;  
using System.Text;
```

```
namespace OutlookAddin4  
{  
    class ThisApplication  
    {  
    }  
}
```

Aby sme zaistili viditeľnosť triedy v celom projekte, do zdrojového kódu novovytvorenej triedy pridáme kľúčové slová `public partial`. Do deklarácie namespace pridáme kód

```
using System.Windows.Forms;  
using Office = Microsoft.Office.Core;  
using Outlook = Microsoft.Office.Interop.Outlook;
```

Pre pridanie novej položky menu potrebujeme určiť jej polohu. Bolo by jednoduché umiestniť novú položku ako poslednú, no odporuje to ustáleným zvyklostiam. Poslednou položkou menu býva predsa nápoveda. Preto našu novú položku umiestnime ako predposlednú pred položku nápovedy.

Tu narazíme na drobný problém s lokalizáciou. V anglickej verzii sa táto položka nazýva „Help“, v slovenskej zasa „Pomocník“. Aby nielenže nedošlo k chybe ak „netrafíme“ verziu, je výhodné túto časť zapúzdriť do programovej konštrukcie pre ošetrenie výnimky. Ak sme správne určili názov položky, bude nová položka pridaná pred ňu. Tento postup je univerzálny, takže novú položku môžeme umiestniť kamkoľvek. V opačnom prípade zistíme počet položiek a novú položku umiestnime ako predposlednú explicitným nastavením hodnoty indexu na hodnotu počtu položiek.

```
try
{
    _helpMenuIndex = _menuBar.Controls[„Pomocník“].Index;
}
catch (ArgumentException)
{
    _helpMenuIndex = _menuBar.Controls.Count;
}
```

kompletný kód triedy pre pridanie položky menu bude

```
public partial class ThisApplication
{
    private Office.CommandBar _menuBar;
    private object _helpMenuIndex;
    private Office.CommandBarPopup _topMenu;

    public void InitializeMenu2()
    {
        _menuBar = this.ActiveExplorer().CommandBars.ActiveMenuBar;

        try
        {
            _helpMenuIndex = _menuBar.Controls[„Pomocník“].Index;
        }
        catch (ArgumentException)
        {
            _helpMenuIndex = _menuBar.Controls.Count;
        }

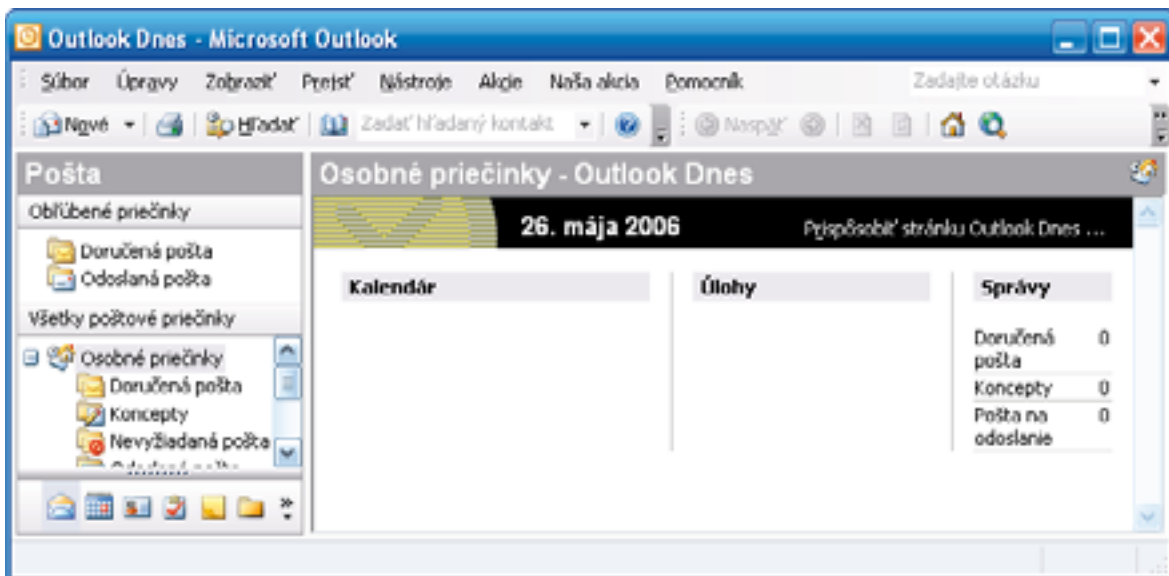
        _topMenu = (Office.CommandBarPopup)_menuBar.Controls.Add(Office.MsoControlType.msoControlPopup, Type.Missing, Type.Missing, _helpMenuIndex, true);
        _topMenu.Caption = „Naša akcia“;
        _topMenu.Visible = true;
    }
}
```

Ak by sme teraz spustili program, doplnok by sa síce nainštaloval, no položka menu by pridaná nebola, nakoľko sme nedefinovali udalosť pri ktorej sa vytvorí inštancia triedy. Toto je tá pravá príležitosť pre využitie udalosti „Startup“.

```
private void ThisApplication_Startup(object sender, System.EventArgs e)
{
    this.InitializeMenu2();
}
```

Po spustení aplikácie si môžeme overiť, že do lišty hlavného menu pribudla nová položka „Naša akcia“ a táto položka je situovaná na správnom mieste, ako predposledná, pred položkou „Pomocník“. Ak sa ju však pokúsime aktivovať, zistíme, že sa zatiaľ rozvinie len prvá prázdna položka.





Outlook8.bmp Rozšírenie menu

V budovaní funkčného menu budeme pokračovať vybudovaním položky roletového menu. Do triedy ThisApplication pridáme deklaráciu objektu CommandBarButton.

```
private Office.CommandBarButton _polozkaMenu;
```

do metódy pre inicializáciu menu pridáme kód pre inicializáciu položky (alebo viacerých položiek) roletového menu

```
_polozkaMenu = (Office.CommandBarButton)_topMenu.Controls.Add(
    Office.MsoControlType.msoControlButton, Type.Missing, Type.Missing, Type.Missing, true);
_polozkaMenu.Caption = „Oznam“;
_polozkaMenu.Visible = true;
_polozkaMenu.Click += new Microsoft.Office.Core._CommandBarButtonEvents_ClickEventHandler(_polozkaMenu_Click);
```

Všimnite si v kóde vytvorenie nadväznosti udalosti zatlačenia tlačidla na obslužnú procedúru

```
void _polozkaMenu_Click(Microsoft.Office.Core.CommandBarButton Ctrl, ref bool CancelDefault)
{
    MessageBox.Show(„Text oznamu“);
}
```

Kompletný kód triedy pre vytvorenie a zaistenie funkcie menu bude

```
public partial class ThisApplication
{
    private Office.CommandBar _menuBar;
    private object _helpMenuIndex;
    private Office.CommandBarPopup _topMenu;
    private Office.CommandBarButton _polozkaMenu;

    public void InitializeMenu2()
    {
        // Get the Outlook menu bar.
        _menuBar = this.ActiveExplorer().CommandBars.ActiveMenuBar;

        try
        {
```

```

        _helpMenuIndex = _menuBar.Controls[„Pomocník“].Index;
    }
    catch (ArgumentException)
    {
        _helpMenuIndex = _menuBar.Controls.Count;
    }

    // hlavne menu
    _topMenu = (Office.CommandBarPopup)_menuBar.Controls.Add(Office.MsoControlType.msoControlPopup, Type.
Missing, Type.Missing, _helpMenuIndex, true);
    _topMenu.Caption = „Naša akcia“;
    _topMenu.Visible = true;

// položka menu
    _polozkaMenu = (Office.CommandBarButton)_topMenu.Controls.Add(
Office.MsoControlType.msoControlButton, Type.Missing, Type.Missing, Type.Missing, true);
    _polozkaMenu.Caption = „Oznam“;
    _polozkaMenu.Visible = true;
    _polozkaMenu.Click += new
Microsoft.Office.Core._CommandBarButtonEvents_ClickEventHandler(_polozkaMenu_Click);
}

void _polozkaMenu_Click(Microsoft.Office.Core.CommandBarButton Ctrl, ref bool CancelDefault)
{
    MessageBox.Show(„Text oznamu“);
}
}

```

## KAPITOLA 9: ZABEZPEČENIE S ŠÍRENIE APLIKÁCIE

Na rozdiel od klasických aplikácií, kde nám stačí vytvoriť príslušné adresáre, nakopírovať výkonné exe súbory, pomocné súbory, prípadne aj súbory s údajmi, pričom na cieľovom počítači musí byť nainštalovaná technologická platforma, inštalácia VSTO aplikácie na cieľový počítač má určité špecifiká. Nakoľko operačný systém je implicitne nastavený na čo najvyššiu úroveň zabezpečenia, budeme musieť pre VSTO aplikácie nastaviť niektoré hodnoty parametrov na požadované hodnoty, alebo jednoduchšie povedané, musíme nastaviť zabezpečenie na úrovni operačného systému tak, aby tento našim VSTO aplikáciám dôveroval a umožnil ich beh.

### Zabezpečenie VSTO aplikácie

Zabezpečenie VSTO aplikácie môžeme rozdeliť na dva okruhy problémov. Na zabezpečenie assembly a zabezpečenie dokumentu. Zabezpečenie assembly (súbor s príponou dll) vychádza primárne zo zabezpečenia technológie .NET Framework 2.0, pomocou ktorej bol assembly súbor vytvorený. Nastavenie zabezpečenia preto vychádza z nastavenia v .NET Security. Štandardné nastavenie pre túto technológiu je také, že sa nedôveruje dokumentom, ktoré nepochádzajú zo zóny Local Machine. Assembly vyžaduje pre svoj beh nastavenie úrovne zabezpečenia na úroveň „Full Trust“. Ak nie je nastavená úroveň „Full Trust“, assembly sa nepripojí a dokument sa bude správať presne tak ako štandardný dokument programu kancelárskeho balíka Office. Napríklad dokument sa zobrazí bez panelu ovládacích prvkov, nebude možné využívať ani funkcionality prvkov umiestnených priamo na ploche dokumentu.

V procese nasadenia VSTO aplikácie sa využívajú dva manifesty:

- aplikačný manifest
- manifest nasadenia

Aplikačný manifest je súčasťou dokumentu a obsahuje informáciu o príslušnom assembly súbore, prípadne dokumentuje ďalšie závislosti. Obsahuje taktiež informáciu, kde sa nachádza manifest nasadenia.

Pre aplikáciu WordDocument1 bude aplikačný manifest v súbore WordDocument1.application

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1 assembly.adaptive.xsd"
manifestVersion="1.0" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xmlns="urn:schemas-microsoft-com:asm.v2"
xmlns:asmv1="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-microsoft-com:asm.v2" xmlns:
xrml="urn:mpeg:mpeg21:2003:01-REL-R-NS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <assemblyIdentity name="WordDocument1.application" version="1.0.0.0" publicKeyToken="0000000000000000"
language="neutral" processorArchitecture="msil" xmlns="urn:schemas-microsoft-com:asm.v1" />
  <description asmv2:publisher="LL" asmv2:product="WordDocument1" xmlns="urn:schemas-microsoft-com:asm.v1"
/>
  <deployment install="true" />
  <dependency>
    <dependentAssembly dependencyType="install" codebase="WordDocument1_1.0.0.0\WordDocument1.dll.
manifest" size="729">
      <assemblyIdentity name="WordDocument1.dll" version="1.0.0.0" />
      <hash>
        <dsig:Transforms>
          <dsig:Transform Algorithm="urn:schemas-microsoft-com:HashTransforms.Identity" />
        </dsig:Transforms>
        <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <dsig:DigestValue>iBBiJHdweI6Z58xEueUJgnNHXgM=</dsig:DigestValue>
      </hash>
    </dependentAssembly>
  </dependency>
</asmv1:assembly>
```

Manifest nasadenia bude v súbore WordDocument1.dll.manifest

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-microsoft-com:asm.v2"
manifestVersion="1.0">
  <assemblyIdentity name="WordDocument1.dll" version="1.0.0.0" />
  <asmv2:entryPoint name="Startup" dependencyName="dependency0">
    <asmv2:clrClassInvocation class="WordDocument1.ThisDocument" />
  </asmv2:entryPoint>
  <asmv2:dependency asmv2:name="dependency0">
    <asmv2:dependentAssembly>
      <assemblyIdentity name="WordDocument1" version="1.0.0.0" />
    </asmv2:dependentAssembly>
    <asmv2:installFrom codebase="WordDocument1_1.0.0.0\WordDocument1.dll" />
  </asmv2:dependency>
  <asmv2:installFrom codebase="file:///c:/deploy/aplikacia/WordDocument1.application" />
</assembly>
```

Aplikácia si pri spustení zistí z aplikačného manifestu informáciu o tom kde sa nachádza manifest nasadenia. V manifeste nasadenia si overí či verzia assembly ktorú používa je zhodná z aktuálnou verziou na serveri. Na základe takto získaných informácií sa ak je to potrebné stiahne zo serveru nová verzia assembly.

Zabezpečenie dokumentu vychádza zo zabezpečenia miesta jeho uloženia

## Šírenie VSTO aplikácie

Pre nasadenie VSTO aplikácie sú k dispozícii tri scenáre nasadenia

- Local / Local
- Local / Network
- Network / Network

**Local/Local** scenár, kedy dokument aj assembly sú na lokálnom počítači nevyžaduje pripojenie na server ani pri otváraní dokumentu. Na druhej strane administrátor musí nastaviť na lokálnom stroji vhodnú bezpečnostnú politiku, to znamená, že musí pre používané „assembly“ nastaviť stupeň dôveryhodnosti na úroveň „FullTrust“. Pri aktualizácii je potrebná redistribúcia aplikácie na všetky počítače, na ktorých je aplikácia nasadená.

Konfigurácia **Local/Network**, kedy dokument je na lokálnom počítači a assembly na serveri umožňuje presonalizáciu dokumentov a ich centrálnu aktualizáciu. Pri tomto scenári nie je dôležité umiestnenie dokumentov. Nevýhoda nutnosti nastavovania bezpečnostnej politiky administrátorom na lokálnom počítači zostáva podobne ako v predchádzajúcom scenári. Pri otváraní dokumentu je u scenára Local/Network nevyhnutné sieťové pripojenie na server.

Scenár nasadenia **Network/Network** prináša so sebou možnosť centrálny vykonávaných aktualizácií či už dokumentu, alebo assembly. Ak sú dokumenty umiestnené na serveri je možné ich zdieľanie v rámci tímovej spolupráce. Tento scenár vyžaduje trvalé sieťové pripojenie.

## Príprava klientského počítača

Skôr než začneme šíriť VSTO aplikáciu z vývojárskeho počítača, alebo serveru na lokálne klientské počítače, na ktorých tieto aplikácie pobežia je potrebné na nich vytvoriť vhodné prostredie pre ich beh. Je potrebné nainštalovať tieto podporné komponenty a aplikácie

- technologickú platformu .NET Framework 2.0
- Microsoft Office 2003, prípadne samostatný program z tohto balíka (Word, Excel...) pre ktorý je VSTO aplikácia vytvorená, vo verzii minimálne so Service Packom SP1
- Visual Studio Tools For Office Runtime
- Office Primary Interop Assemblies

### .NET Framework 2.0

Základným pilierom pre inštaláciu VSTO aplikácie na cieľový počítač je technologická platforma .NET Framework 2.0

### Microsoft Office SP1

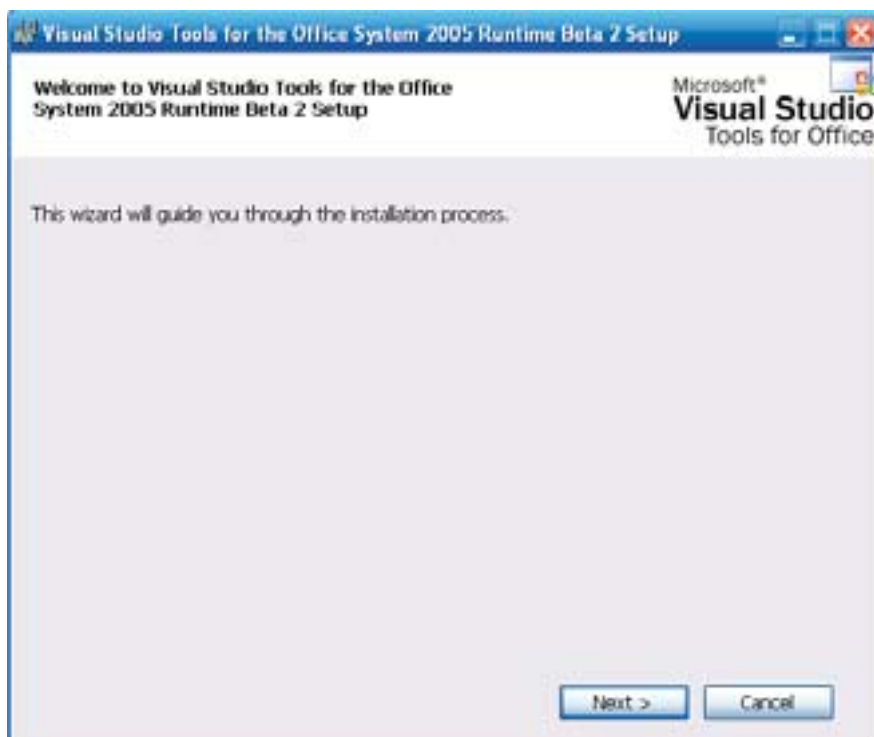
Na cieľovom počítači musí byť nainštalovaný kancelársky balík Microsoft Office, prípadne niektorý z programov tvoriacich tento balík, napríklad:

- Microsoft Office Professional Edition 2003.
- Microsoft Office Professional Enterprise Edition 2003.
- Microsoft Office Excel 2003.
- Microsoft Office Word 2003.
- Microsoft Office Outlook 2003.
- Microsoft Office Professional Edition 2003 Trial.
- Microsoft Office System Evaluation 2003 Enterprise Edition.

Samozrejme vo vzťahu k tomu pre aký program budeme používať VSTO aplikácie. Kombinácia VSTO aplikácie pre Word a na klientskom počítači mať nainštalovaný len program Excel 2003 je nezmyselná. Service Pack 1 je nevyhnutnou podmienkou

### Visual Studio Tools for Office Runtime

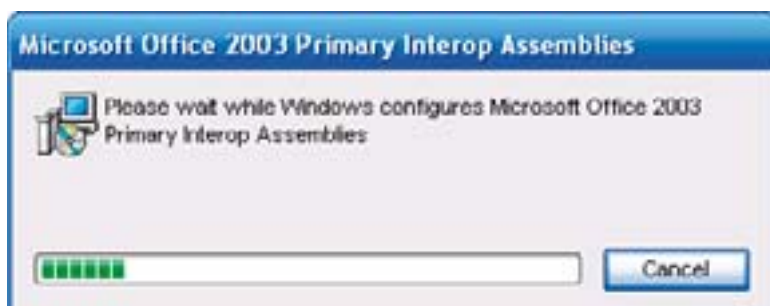
Tretím nevyhnutným predpokladom pre beh VSTO programov na klientskom počítači je súbor knižníc VSTOR. Inštaluje sa ako jediný inštalačný súbor o veľkosti 1.33 megabajtov. Súbor VSTOR.EXE je k dispozícii na adrese (<http://go.microsoft.com/fwlink/?linkid=49612&clcid=0x409>)



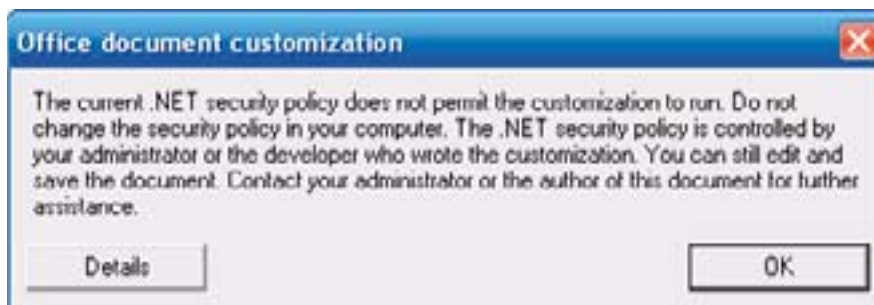
*Inštalácia Visual Studio Tools for Office Runtime na cieľový počítač*

## **Office Primary Interop Assemblies.**

Poslednou položkou inštaláčného procesu je Office Primary Interop Assemblies. Je to vrstva rozhraní medzi technológiou .NET a COM objektmi Office



*Úvodný dialóg sprievodcu „Publishing Wizard“*



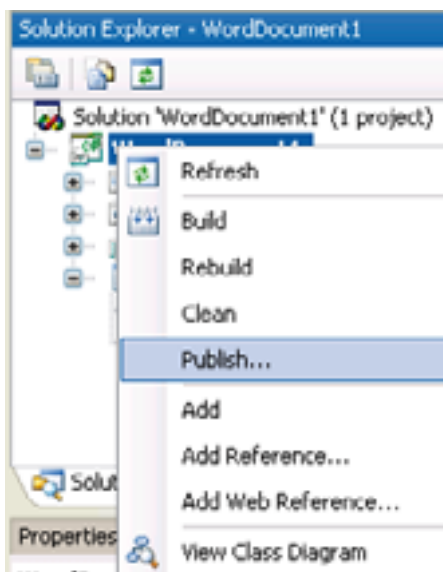
*Upozornenie na prístupové privilégia*

## Microsoft.NET Framework 2.0 SDK (voliteľné)

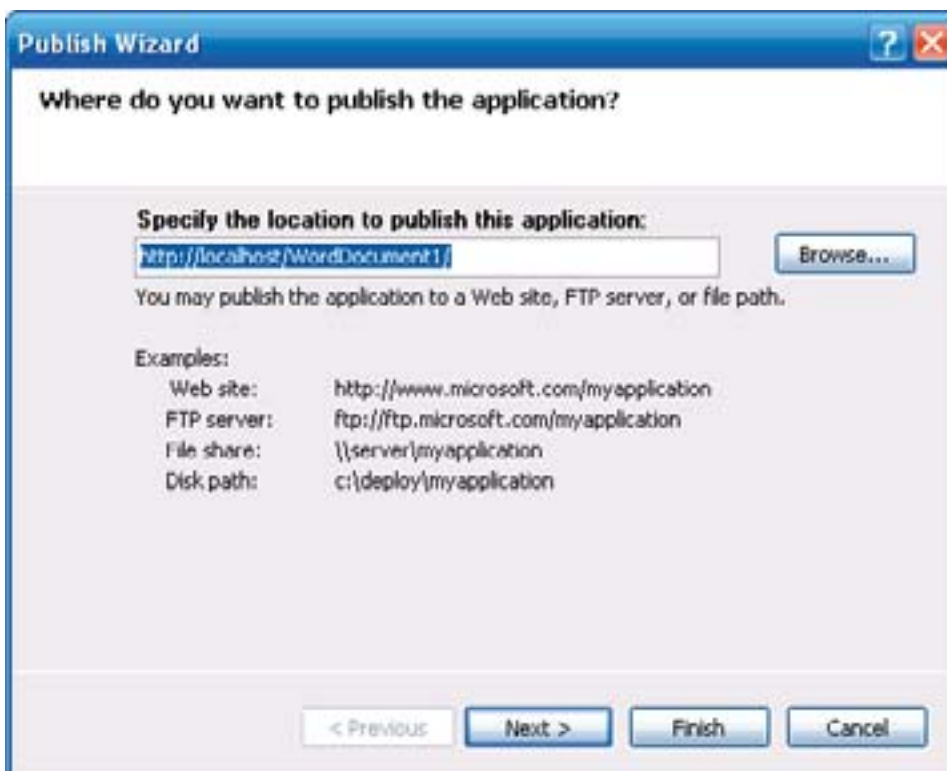
Pri konfigurácii klientského počítača budeme potrebovať nastaviť stupeň dôveryhodnosti DLL assembly na hodnotu „Full Trust“. Najjednoduchšie tento úkon vykonáme pomocou nástroja Microsoft.NET Framework 2.0 Configuration. Táto inštalácia je voliteľná nakoľko dôveryhodnosť assembly môžeme nastaviť aj pomocou nástroja Caspol, ktorý je súčasťou štandardného distribučného balíka.NET Framework Redist.

## Šírenie aplikácie pomocou nástroja Publishing Wizard

Publishing Wizard aktivujeme pomocou položky „Publish“ kontextového menu v okne „Solution Explorer“. Filozofia jeho fungovania je rovnaká ako pri šírení Win Forms aplikácií.



Aktivovanie položky kontextového menu Publish



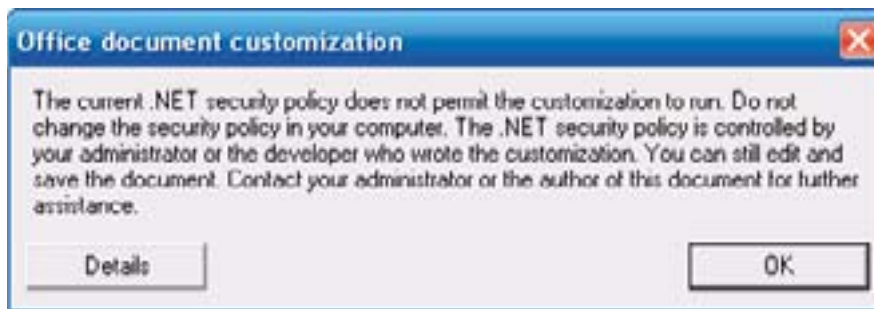
Úvodný dialóg sprievodcu „Publishing Wizard“



Aplikáciu môžeme šíriť buď prostredníctvom webového sídla, môžeme ju umiestniť na FTP server, prípadne do zdieľaného adresára, alebo do adresára na disku. V druhom a zároveň poslednom kroku vytvorí sprievodca „Publish Wizard“ inštalačné súbory.

## Nastavenie úrovne zabezpečenia pre VSTO aplikáciu

Ak by sme sa po vykonaní prípravných krokov a prekopírovaní alebo nainštalovaní aplikácie do cieľového adresára pokúsili VSTO aplikáciu spustiť, naše úsilie zmarí chybové hlásenie, alebo presnejšie oznam o nedostatočnom nastavení bezpečnostnej politiky lokálneho počítača.



Oznam o zle nastavenej politike dôveryhodnosti

Aby sme mohli spúšťať VSTO aplikáciu z príslušného adresára na klientskom počítači je potrebné nastaviť úroveň zabezpečenia, presnejšie úroveň dôveryhodnosti kódu príslušnej DLL assembly, prípadne aplikačného adresára v ktorom sa nachádza viac assembly súborov na hodnotu „Full Trust“. Nastavenie najjednoduchšie vykonáme pomocou nástroja Microsoft.NET Framework 2.0 Configuration. Drobný problém je v tom, že tento nástroj nie je súčasťou „redist“ distribúcie technologickej platformy.NET Framework. Môžeme ho však nainštalovať spolu s balíkom Microsoft.NET Framework 2.0 SDK.

Druhá možnosť je nastavenie dôveryhodnosti pomocou nástroja Caspol, ktorý je súčasťou štandardného distribučného balíka.NET Framework Redist.



Spustenie nástroja Microsoft.NET Framework 2.0 v menu „Control Panel“, zložke „Administrative tools“

Po spustení nástroja .NET Framework 2.0 Configuration rozvineme v jeho ľavom okne hierarchickú stromovú štruktúru. V zložke „Runtime Security Policy“ nájdeme tri podzložky predstavujúce úrovne nastavenia.

- Enterprise
- Machine
- User

Na úrovni Enterprise nastavujeme príslušné oprávnenia alebo dôveryhodnosť pre celú lokálnu sieť. Úroveň Machine predstavuje príslušný klientský počítač. Najnižšiu úroveň nastavenia predstavuje User, kedy nastavujeme oprávnenia len pre konkrétneho používateľa, napríklad ak je klientský počítač pripojený do domény. Na každú úroveň okrem najnižšej úrovne User musíme byť prihlásení ako administrátor príslušnej úrovne. Úroveň nastavenia volíme podľa aplikovanej politiky nastavenia a zabezpečenia počítačov a siete.



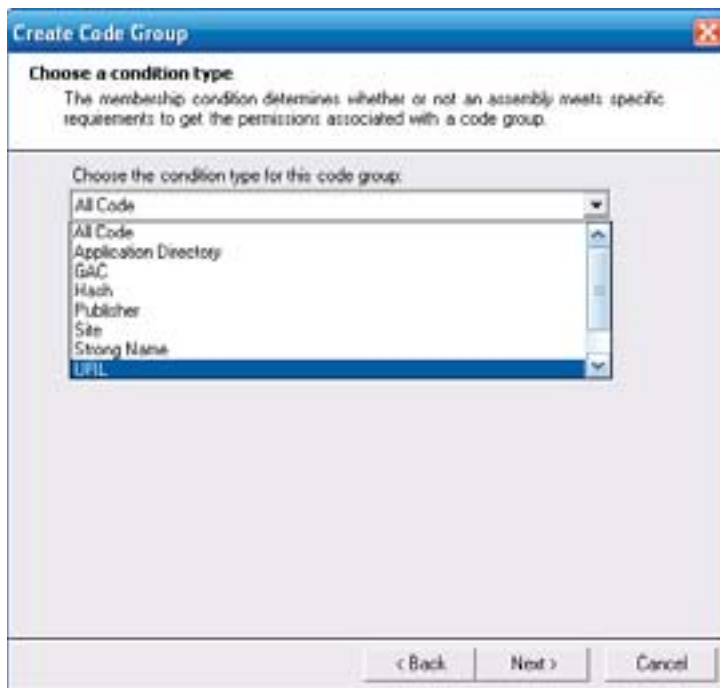
Microsoft.NET Framework 2.0

Pre ilustráciu nastavíme úroveň dôveryhodnosti pre celý adresár VSTO\_POKUSY a to na úrovni konkrétneho aktuálneho používateľa. V zložke User označíme podzložku All\_Code. V pravom okne sa zobrazia dva odkazy: „Edit Code Group Property“ a „Add a Child Code Group“. Pre naše účely nastavenia vyberieme link „Add a Child Code Group“, pomocou ktorého vytvoríme kódovú podskupinu VSTO



Vytvorenie kódovej podskupiny

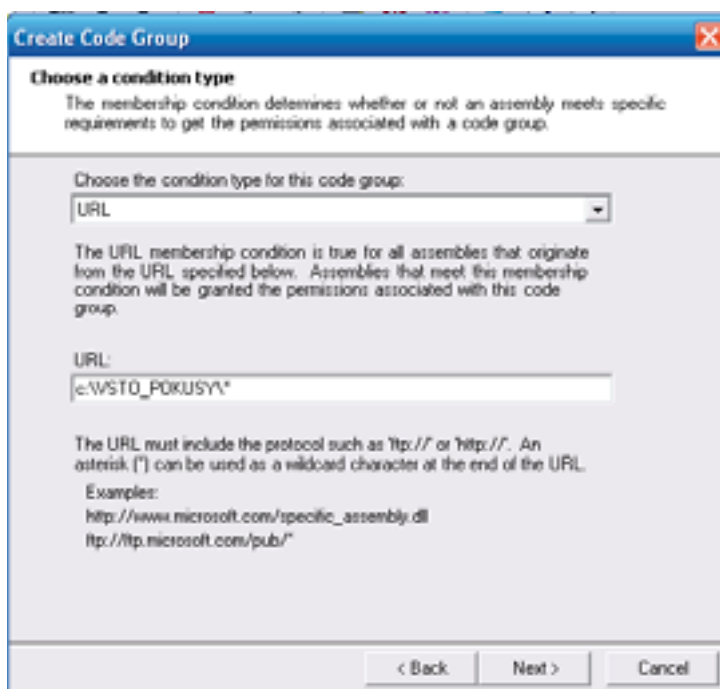
V dialógu „Create Code Group“ zadáme názov kódovej podskupiny, napríklad VSTO. Názov kódovej skupiny sa nemusí zhodovať s názvom objektu, napríklad adresára na ktorý sa aplikuje. Tlačidlom „Next“ postúpime do ďalšieho dialógu pre výber objektu na ktorý sa budú nastavené pravidlá bezpečnostnej politiky vzťahovať. Pre bežný adresár vyberieme voľbu URL.



*Výber objektu na ktorý sa nastavené pravidlá aplikujú*

Do nasledujúceho dialógu zadáme názov URL. Ak chceme danú politiku aplikovať na klasický adresár a jeho podadresáre zadávame názov adresára ukončený hviezdíčkou, v našom prípade konkrétne

*c:\VSTO\_POKUSY\\*. Ak chceme zadať URL adresu len pre konkrétnu DLL assembly, zadávame ju v tvare napríklad c:\<path>\ExcelApplication1.dll alebo c:\<path>\ExcelApplication1\bin\\**



*Zadanie adresára pre nastavené pravidlá*



Nastavenie stupňa dôveryhodnosti pre vybraný adresár

Finálnym krokom je nastavenie stupňa dôveryhodnosti. Pre VSTO aplikácie je potrebné nastaviť úroveň Full Trust.

## Aplikácia CASPOL

Aplikáciu CASPOL nájdeme v pracovnom adresári príslušnej verzie .NET Frameworku, v našom prípade konkrétne v adresári

`c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\`

Nápovedu k aplikácii získame pomocou príkazu `caspol -?`

```
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>caspol -?
Microsoft (R) .NET Framework CasPol 2.0.50727.42
Copyright (c) Microsoft Corporation. All rights reserved.
Syntax: caspol <option> <args> ...
```

```
caspol -m[achine] nastavenie úrovne pre tento počítač
caspol -u[ser] nastavenie úrovne pre tohto používateľa
caspol -en[terprise] nastavenie enterprise úrovne
caspol -cu (-customuser) <path> nastavenie vlastnej úrovne
caspol -a[ll] nastavenie politiky pre všetky aktívne úrovne
caspol -ca (-customall) <path> všetky úrovne pre daného používateľa
caspol -l[list] Zoznam kódových skupín a úrovní nastavenia
caspol -lg (-listgroups) Zoznam kódových skupín
caspol -lp (-listpset) Zoznam úrovní nastavenia
caspol -lf (-listfulltrust) zoznam assemblies pre úroveň full frust
caspol -ld (-listdescription) popis kódových skupín
caspol -cft (-checkfulltrust) zoznam oprávnení full trust
```

```
caspol -ap (-addpset) { <named_xml_file> | <xml_file> <name> } pridanie pomenovanej úrovne
caspol -cp (-chgpset) <xml_file> <pset_name> zmena nastavení aktívnej úrovne
caspol -rp (-rempset <pset_name>) odstránenie nastavenia prístupovej úrovne
caspol -af(-addfulltrust) <assembly_name> nastavenie úrovne úplnej dôvery
```

*caspol -rf (-remfulltrust) <assembly\_name> zrušenie úrovne úplnej dôvery*  
*caspol -rg (-remgroup) <label|name> zrušenie kódovej skupiny*  
*caspol -cg (-chggroup) <label|name> {<mship>|<pset\_name>|<flag>}+*  
*zmena kódovej skupiny pre <label|name>*  
*caspol -ag*  
*caspol -addgroup <parent\_label|name> <mship> <pset\_name> <flag>*  
*Add code group to <parent\_label|name> with given membership,*  
*permission set, and flags*

*caspol -rsg (-resolvegroup) <assembly\_name> zoznam príslušných kódových skupín*  
*caspol -rsp (-resolveperm) <assembly\_name> zoznam oprávnení pre súbor*  
*caspol -s[security] { on | off } Zapnutie / vypnutie bezpečnostnej politiky*  
*caspol -e[execution] { on | off } Povolenie alebo zákaz oprávnení pre spustenie*  
*caspol -pp (-polchgprompt) { on | off } povolenie alebo zákaz promptu pri zmenách*  
*caspol -q[uiet] zákaz promptu pre tento príkaz*  
*caspol -r[ecover] - obnovenie uloženej verzie*  
*caspol -rs (-reset) reset na implicitné hodnoty nastavenie*  
*caspol -rsld (-resetlockdown) reset na implicitný lockdown stav*  
*caspol -b[uildcache] povolenie vytvorenia cache pre bezpečnostnú politiku*  
*caspol -? (/?) (/?) nápoveda*

kde „<mship>“ môže byť:

- allcode Všetok kód
- appdir aplikačný adresár
- custom <xml\_file> vlastné podmienky
- hash <hashAlg> {-hex <hashValue>|-file <assembly\_name>} Assembly hash
- pub {-cert <cert\_file\_name> | -file <signed\_file\_name> | -hex <hex\_string>}  
Distribútor sw
- gac inštalácia Global Assembly Cache
- site <website> webové sídlo
- strong {-file <assemblyfile\_name> | -hex <public\_key>}  
{<name> | -noname} {<version> | -noverion} Strong name
- url <url> URL
- zone <zone\_name> zóna môže byť: MyComputer, Intranet, Trusted, Internet, Untrusted

kde „<flag>“ môže byť kombináciou uvedených položiek:

- exclusive {on|off} nastav príznak politiky Exclusive
- levelfinal {on|off} nastav príznak politiky LevelFinal
- n[ame] <name> názov Code group
- d[escription] <desc> popis Code group

Pre VSTO aplikáciu môžeme nastaviť úroveň Full Trust príkazom

*caspol.exe -q -u -ag 1 -url \"adresa\" FullTrust -n \"adresa\"*